UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: Servicios Bluetooth para espacios públicos (Blue4AS)
Autores: Barrientos Leandro Nicolás, Zozaya Juan Esteban
Director: Gordillo Silvia
Carrera: Licenciatura en Sistemas

Resumen

El rápido avance producido en los últimos 5 años por las tecnologías de hardware y comunicaciones hacen cada vez más factible la creación de aplicaciones ubicuas – capaces de adaptarse a la ubicación, contexto, dispositivo de uso y perfil del usuario– mediante las cuales diferentes tipos de usuarios colaboran activamente para realizar sus tareas. La proliferación de dispositivos móviles y la pervasividad de Internet, y en particular de la Web, hacen de este un escenario de uso cotidiano. Blue4AS es un prototipo basado en la comunicación vía Bluetooth con sensores diseminados que brinda a edificios o espacios multitudinarios y de alta circulación, la capacidad de transmitir información referente al lugar en donde se encuentre el usuario mediante su dispositivo móvil y los sensores Bluetooth, así como también permite adquirir conocimiento por parte de quien lo implemente. Escenarios posibles de implementación son: escuelas, facultades, museos, zoológicos, galerías de arte, edificios históricos y paseos culturales, etc.

Palabras Claves

Bluetooth, Proximity Marketing, Context Aware, Computación Ubicua, Sistemas de Información Móvil, Software Libre, Object Push, Sockets, RFCOMM, SDP, Accesibilidad.

Trabajos Realizados

Este trabajo fue realizado con el objetivo de orientar a las personas en espacios públicos con herramientas de bajo costo utilizando tecnología Bluetooth y software libre. El prototipo desarrollado cuenta con dos módulos: uno de administración remota, que permite el monitoreo y la configuración (red de sensado, zonas, datos a enviar, etc.), y otro, el sistema ubicuo, que se encarga de realizar la comunicación con los dispositivos. Se logró un servicio de múltiples propósitos, sensible al contexto y con una mayor participación e incidencia en el usuario (incluyendo aquellos con capacidades diferentes)

Conclusiones

El prototipo fue puesto en producción en un entorno real en la Facultad de Ciencias Jurídicas y Sociales de la UNLP. Durante la etapa de pruebas, su funcionamiento ha superado las expectativas, demostrando que el mismo se puede poner en producción con total éxito y aplicándolo a múltiples escenarios. Esta implementación inicial permitió, por primera vez en su historia, obtener estadísticas sobre diferentes flujos de circulación, permanencia por sectores, estimación de franjas horarias de mayor y menor circulación, cantidad de ingresos por su acceso principal, cantidad de envíos exitosos, etc.

Trabajos Futuros

- *Integración con nuevas tecnologías y diferentes niveles de seguridad (protocolos autenticados y emparejamiento automático)*
- *Módulos Interactivos: vías bidireccionales de comunicación, interacción activa del usuario*
- *Aplicaciones Clientes para SmartPhones*

Fecha de la presentación: NOVIEMBRE 2011

SERVICIOS BLUETOOTH PARA ESPACIOS PÚBLICOS

Alumnos:

Barrientos, Leandro Nicolás

Zozaya, Juan Esteban

Directora:

Gordillo, Silvia

Facultad de Informática

Universidad Nacional de La Plata

Índice

Introducción....6

Organización de este trabajo....8

1 Sistemas de Información Móvil....11

1.1 Impacto de los Dispositivos Móviles....11

1.2 Acceso del usuario....12

1.3 Ventajas y desventajas....12

1.4 Computación Ubicua y Sensible al Contexto....13

1.5 Frameworks Context-Aware....17

2 Tecnología Bluetooth....22

2.1 Clases y Frecuencias....22

2.2 Especificaciones....23

2.2.1 Radio Bluetooth:....24

2.2.2 Banda Base:....25

2.2.3 LMP (Link Manager Protocol):....25

2.2.4 HCI (Host Controller Interface):....25

2.2.5 L2CAP (Logical Link Control and Adaption Protocol):....25

2.2.6 RFCOMM (Radio Frequency Communications):....26

2.2.7 SDP (Service Discovery Protocol):....26

2.3 Perfiles....27

2.3.1 General Access Profile (GAP):....29

2.3.2 Serial Port:....29

2.3.3 Generic Object Exchange (GOEP):....29

2.4 Redes Bluetooth: Piconets y Scatternets....30

2.5 Comunicación entre los dispositivos....31

2.6 Búsqueda de Información de Servicios....33

2.7 Comunicación vía Sockets....35

2.8 Limitaciones....36

2.9 Seguridad....37

2.10 Frecuencias, Bandas, potencias y estudios sobre la salud....40

2.10.1 Banda ISM....41

2.10.2 Estudios sobre la salud, recomendaciones de la OMS (organización Mundial de la Salud)....42

2.10.3 Recomendaciones de la OMS....42

3 Software Libre....44

3.1 Copyleft....45

3.2 Categorías de software....46
3.2.1 Software Libre:....46
3.2.2 Software de Código Abierto (Open Source):....47
3.2.3 Software de Dominio Público....47
3.2.4 Software protegido con copyleft....47
3.2.5 Software libre no protegido con copyleft (tipo XFree86)....47
3.2.6 Software cubierto por la GPL....48
3.2.7 Software Privativo....48
3.2.8 Software Cerrado....48
3.2.9 Shareware....48
3.3 Plataforma Linux....49
3.3.1 Distribuciones....50
3.4 Python....50
3.4.1 Características del lenguaje....51
3.4.2 Extensión para Bluetooth....51
3.4.3 Persistencia....52
3.5 Django....52
3.5.1 Características Principales....53
4 Proyecto Blue4AS (Bluetooth For Accessibility Systems)....54
4.1 Motivación....54
4.2 Desarrollo propuesto....55
4.3 Requerimientos....55
4.4 Arquitectura utilizada....56
4.5 Aplicación....58
4.5.1 Escenario típico....58
4.5.2 Sistema Ubicuo....58
4.5.2.1 Funcionamiento....58
4.5.2.2 Políticas de broadcasting (Blacklist o Whitelist)....63
4.5.2.2.1 Blacklist....64
4.5.2.2.2 Whitelist....64
4.5.2.2.3 Mecanismo de suscripción....64
4.5.2.3 Modo debugging....66
4.5.2.4 Detalles de la implementación (detalles técnicos y definición de clases)....67
4.5.2.4.1 Modificaciones sobre la librería PyBluez (bluez.py)....82
4.5.3 Administración Remota....84
4.5.3.1 Introducción....84
4.5.3.2 Funcionamiento....84
4.5.3.3 Configuración del sistema....85

4.5.3.4 Detalles de la implementación....87
4.5.3.4.1 Formato vCard: 90
4.5.3.4.2 Formato iCalendar....92
4.6 Decisiones Tomadas....98
4.7 Performance....99
5 Otras aplicaciones y trabajos futuros....101
5.1 Otras aplicaciones....101
5.1.1 Módulo de mensajes de emergencia (con manejo de prioridades) ante siniestros, desperfectos edilicios y/o falta de servicios básicos....101
5.1.2 Seguimiento personalizado....103
5.1.3 Envío de información según el idioma del usuario....103
5.1.4 Envío de información selectiva por dispositivos....104
5.2 Trabajos futuros....105
5.2.1 Integración con nuevas tecnologías y diferentes niveles de seguridad....105
5.2.2 Módulos Interactivos....106
5.2.3 Aplicaciones Clientes para SmartPhones....107
5.2.4 Desarrollo de una capa intermedia entre el modelo de objetos y de datos....108
6 Experiencia....110
6.1 Cálculos estimados sobre la implementación....110
6.2 Pruebas de campo....111
6.2.1 ¿Por qué la elección de cada lugar? 112
6.2.2 Puesta a punto....113
6.2.2.1 Configuración inicial....113
6.2.2.2 Política de Broadcast....113
6.2.2.3 Datos a enviar....113
6.2.3 Problemas encontrados....116
6.2.3.1 Problemas de software....116
6.2.3.2 Problemas de hardware....117
6.2.3.2.1 Problemas de hardware defectuoso 117
6.2.3.2.2 Problemas de sensores....117
6.3 Datos relevados....122
6.3.1 Dispositivos únicos y recurrentes....122
6.3.2 Promedio de dispositivos por día....123
6.3.3 Cantidad de dispositivos por tiempo de permanencia....125
6.3.4 Cantidad de envíos totales....127
6.3.5 Datos enviados por lugar....129
6.3.6 Datos enviados por tipo de dato....130
6.3.7 Dispositivos detectados por hora....130

6.3.8 Dispositivos detectados por fecha....132
6.3.9 Dispositivos detectados por su tipo....133
6.3.10 Dispositivos detectados por lugar....134
6.3.11 Nombres de los dispositivos....136
6.3.12 Análisis de los datos relevados....137
7 Conclusiones....138
8 Bibliografía....144

Introducción

El rápido avance producido en los últimos 5 años por las tecnologías de hardware y comunicaciones hacen cada vez más factible la creación de aplicaciones ubicuas –es decir, capaces de adaptarse a la ubicación, contexto, dispositivo de uso y perfil del usuario– mediante las cuales diferentes tipos de usuarios colaboran activamente para realizar sus tareas. La proliferación de dispositivos móviles y la pervasividad de Internet, y en particular de la Web, hacen de este un escenario de uso cotidiano.

En la actualidad, la mayoría de los dispositivos móviles tienen incorporada tecnología Bluetooth, desde teléfonos celulares hasta tabletas, e-readers, reproductores de música y video, y computadoras portátiles. Una de las características más destacadas de esta tecnología es el funcionamiento por cercanía, lo cual resultó decisivo a la hora de su elección como canal de comunicación primario en este proyecto. Esta tecnología es utilizada por lo general para múltiples propósitos, siendo los más frecuentes la transferencia de información entre diferentes dispositivos, la conectividad de auriculares, uso de manos libres en los automóviles y manejo remoto de dispositivos, sin necesidad de conectarlos mediante cables.

Los recursos que ofrece la tecnología se encuentran poco explotados, ya que si bien permite el manejo de dispositivos cercanos sin necesidad de una conexión cableada entre ellos, no es algo que esté totalmente masificado ni utilizado con otros objetivos que no sean el de la transferencia de archivos y la administración remota. El Bluetooth no está pensado para la cooperación de sensores con varios fines en común, ya sea para esparcir información específica por zonas, recolectar datos para generar estadísticas de seguimiento y conocimiento del tráfico, y gracias a ello, tomar decisiones edilicias. Tampoco está pensado o explotado totalmente para el turismo, y menos aún para la inclusión de personas que no posean todas las capacidades sensoriales. Generalmente, todas estas aplicaciones, si se encuentran desarrolladas, lo están con equipamiento específico, altos costos y software cerrado.

Existen diferentes desarrollos que se aproximan a resolver parcialmente dicha problemática. Son las denominadas aplicaciones de Marketing de Proximidad, las cuales se basan en el envío de información publicitaria en – por lo general - centros y eventos comerciales. Si bien admiten diferentes formatos de emisión, no cuentan con información estadística, discriminación por zonas, adaptación al hardware, políticas de transferencia, etc.

En el presente trabajo se presenta un prototipo basado en la comunicación vía Bluetooth de sensores diseminados que brinda a edificios o espacios multitudinarios y de alta circulación, la capacidad de transmitir información referente al lugar en donde se encuentre el usuario mediante su dispositivo móvil y los sensores Bluetooth. Entre los lugares de aplicación de este prototipo se encuentran escuelas, facultades, museos, zoológicos, galerías de arte, edificios históricos y paseos culturales, etc. Este prototipo se desarrolló con la idea de generar una aplicación de múltiples propósitos, sensible al contexto y con una mayor participación e incidencia en el usuario (incluyendo aquellos con capacidades diferentes) que los sistemas tradicionales. Por tal motivo, se incorporó la capacidad de transmitir múltiples formatos (imagen, audio, eventos, contactos y texto). El prototipo es capaz de adaptarse a diversos escenarios, y al correr sobre un computador con bajos recursos, no requiere de un hardware específico ni sofisticado. A su vez admite múltiples sensores conectados a él y se puede variar dinámicamente la configuración de los mismos; cuantos más sensores tenga, más eficiente será la detección y transferencia, gracias al balanceo de carga.

La implantación de este prototipo se llevó a cabo en la Facultad de Ciencias Jurídicas y Sociales de la Universidad Nacional de La Plata. Esta facultad posee un tránsito muy elevado de personas, lo que permitió experimentar con las principales características del sistema. Los usuarios del prototipo fueron precisamente los alumnos, docentes y personal administrativo de dicha dependencia, quienes circularon por las diferentes zonas de detección.

Este prototipo fue desarrollado con el propósito específico de brindar orientación al usuario dependiendo de su ubicación y su entorno mediante el envío de información discriminada por zona. Adicionalmente, se planteó como objetivo el adquirir conocimiento por parte de la Facultad a partir de las detecciones almacenadas y el comportamiento de circulación de los usuarios. Esto último motivó la incorporación de un módulo capaz de registrar eventos y de generar estadísticas en tiempo real sobre todo lo acontecido en

presencia de los sensores.

En el siguiente trabajo se presenta una implementación del prototipo real, de bajo costo y con mínimo mantenimiento, que aporta a espacios comunes la capacidad de poder interactuar de diferentes formas con los usuarios a través de sus dispositivos móviles. La solución está pensada para soportar diferentes políticas de emisión y configuraciones flexibles basándose en el principio de que la información debe ser accesible para todos. Es por todo lo anteriormente mencionado que se desarrolló una solución en donde personas con capacidades diferentes, mediante un dispositivo acorde a las mismas, pueda acceder a la información que se desea emitir.

Organización de este trabajo

Capítulo 1: *Sistemas de Información Móvil*

En este capítulo se menciona el crecimiento de la telefonía móvil en los últimos años y de las comunicaciones en general, dando lugar a nuevos impactos, conceptos y servicios de accesos al usuario. Luego se realiza una introducción a la computación ubicua y sensible al contexto, y se explican las ventajas de utilizar frameworks *context-aware*, junto con un análisis de las arquitecturas de contexto.

Capítulo 2: Tecnología Bluetooth

En este capítulo se intenta dar una introducción a dicha tecnología para una mayor comprensión del desarrollo propuesto al igual que los servicios y protocolos utilizados.

Se describen las clases y frecuencias utilizadas, así como también la pila de protocolos en cada capa del modelo OSI, detallando las características más relevantes de cada uno de ellos. Se enumeran además los perfiles utilizados, los cuales definen una descripción de la interfaz de comunicación entre dos unidades para un servicio particular.

Por último se mencionan las diferentes formas de comunicación, las diferentes topologías que se pueden formar mediante las redes de sensado (piconets y scatternets), las limitaciones, los alcances, la seguridad y los posibles impactos en la salud de la tecnología.

Capítulo 3: *Software Libre*

En este capítulo se comentan las características y modalidades del software libre, las licencias CopyLeft y las categorías del software, poniendo énfasis en el Software de código abierto (open source) y los diferentes tipos de licencia que los alcanzan.

Se hace una introducción al sistema operativo elegido para el desarrollo y ejecución del sistema (GNU/Linux) y de los lenguajes de programación, librerías y frameworks utilizados para el desarrollo del mismo. En todos los casos se utilizó Software Libre y por eso tienen una mención especial en este capítulo.

Capítulo 4: Proyecto Blue4AS (Bluetooth For Accessibility Systems)

En este capítulo se explica la motivación del proyecto y sus bajos requerimientos para poder llevarlo a cabo, indicando todas las decisiones de modelo e implementación que se tomaron, y mostrando en detalle el funcionamiento del sistema. Se describen detalladamente sus dos módulos (aplicación ubicua y administración remota) indicando la metodología empleada al igual que la comunicación con los dispositivos móviles y la obtención de información útil para la organización. Por último se describen los problemas de performance encontrados y cómo se solucionaron.

Capítulo 5: Otras aplicaciones y trabajos futuros

En este capítulo se describen diferentes aplicaciones del proyecto, indicando las modalidades de funcionamiento bajo circunstancias adaptables. Por otro lado se enumeran ciertos trabajos futuros (o expansiones) y se orienta a una posible solución en base a la investigación realizada. En ambos casos se pueden plantear infinidad de casos, aplicaciones o trabajos que se desprenden del presente dando origen a nuevas ramas de investigación.

Capítulo 6: Experiencia

En este capítulo se evalúa el impacto de la transmisión por Bluetooth en la salud de las personas. Luego se detallan todos los pasos realizados para la puesta a punto del prototipo desarrollado, los problemas encontrados y las soluciones encontradas. Por último se muestran resultados estadísticos realizados a partir de la implantación del sistema en un cierto período de tiempo.

Capítulo 7: Conclusiones

En este último capítulo se realiza un análisis del prototipo a partir de los resultados obtenidos y la experiencia en el desarrollo e implantación del mismo. Se hace hincapié en la relevancia de la accesibilidad a la información transmitida y las consecuencias de implementar el sistema en lugares determinados para poder obtener un mayor conocimiento acerca de la circulación en el recinto y de esta manera promover envío de información detallada a las personas que lo transitan.

1 Sistemas de Información Móvil

El uso de la telefonía móvil en el mundo ha crecido drásticamente en los últimos años, y el desarrollo de los dispositivos móviles afectó a la creación de nuevas aplicaciones basadas en las tecnologías emergentes. Las Tecnologías de Comunicación e Información (ICT por sus siglas en inglés) se refieren a la unión de redes de telefonía con las redes de computadoras mediante un simple sistema de enlace. Estas tecnologías tienden a integrar servicios que tradicionalmente estaban separados, como las redes de computadoras y los medios de comunicación clásicos; y los servicios avanzados de telefonía se están integrando mucho más, como por ejemplo la Red Digital de Servicios Integrados (RDSI) que permite a múltiples servicios de distintos tipos estar disponibles en un único lugar o acceso sin depender de la naturaleza de los datos a transmitir ni del equipo terminal que los genere. [1]

Todo lo mencionado anteriormente conlleva a transformar la manera en que las personas utilizan los recursos de información, debido al crecimiento del ancho de banda en las conexiones y que la misma se de, virtualmente, desde cualquier sitio, lo que abre la puerta a los servicios sensibles al contexto, de los que hablaremos luego.

1.1 Impacto de los Dispositivos Móviles

La capacidad de tener una comunicación permanente, casi independiente de la hora y el lugar del usuario, permite que los mismos puedan crear escenarios no solamente para transmitir voz sino también datos a una alta velocidad. Para esto se debe tener en cuenta la resolución de pantalla del teléfono móvil, su rendimiento y consumo del su procesador, su tamaño de memoria, etc.

Dispositivos tales como *PDAs*[1], teléfonos inteligentes y celulares UMTS (aquellos con tecnología 3G) son los que actualmente permiten navegar por Internet y reproducir música y video. A pesar de sus diferencias técnicas, las aplicaciones móviles se están desarrollando para todos los tipos de dispositivos posibles. Pero aquí lo más importante

1 De sus siglas en inglés, Personal Digital Assistant (Asistente Digital Personal) es una computadora de mano que originalmente fue diseñada como una agenda electrónica con calendario, lista de contactos y recordatorios, y que ahora permite realizar muchas de las funciones de una computadora de escritorio, como por ejemplo navegar por internet, leer correo electrónico, crear documentos, ver películas, etc.

es que los nuevos servicios brindados para el ambiente móvil que logran integrar estas nuevas tecnologías, deben cumplir las expectativas del usuario final de forma tal que se le ofrezca un valor adicional significativo que le sea útil y que posea una interfaz de fácil uso. Al usar dichas tecnologías, se puede ofrecer tanto información básica de diversos temas como información más detallada que le da al usuario un valor adicional, es decir que el sistema le proporcionará vistas simples con una alta especialización en el contenido.

1.2 Acceso del usuario

El acceso a los servicios del sistema de información por parte del usuario es provisto a través de varios canales y dispositivos, que puede utilizarlos en diferentes momentos y contextos de interacción. En los sistemas de información multicanal los usuarios comparten datos y pueden trabajar de forma colaborativa; en cambio en los sistemas de información móviles los servicios y la información pueden variar dependiendo del contexto en que son utilizados. Ahora bien, si además el sistema es adaptable, los requerimientos del usuario pueden variar dinámicamente de muchas maneras y el mismo se debe adaptar inmediatamente a ellos, por lo que deben ser aceptados diferentes niveles de servicios o de paradigmas de interacción entre los sistemas. [2]

1.3 Ventajas y desventajas

La mayor ventaja de este tipo de sistemas de información es su habilidad para proveer nuevos servicios de valor agregado gracias a su movilidad y flexibilidad respecto a su contexto de uso. Y además los servicios pueden estar disponibles mediante conexiones inalámbricas en lugares en los que antes era difícil pensarlo debido a la falta de una infraestructura de red estable y confiable.

También existen algunas limitaciones con aquellos dispositivos que no soportan todos los requisitos necesarios para tener una buena interacción, pero siempre se debe garantizar la accesibilidad y usabilidad de los servicios provistos. El entorno cambia continuamente, lo que fuerza a tener interacciones pequeñas y centradas, y tareas que tienden a ser fragmentadas, no del todo definidas e inmersas en otras actividades.

Al ser una tecnología bastante joven, aún hay limitaciones que hay que tener en cuenta, como ser el roaming, desconexiones frecuentes, agujeros de seguridad, ancho de

banda variable, etc. Todo esto se debe tener muy en cuenta a la hora de diseñar e implementar un sistema móvil confiable.

1.4 Computación Ubicua y Sensible al Contexto

En los inicios de la informática, se solía tener una gran computadora trabajando para muchas personas. Luego hubo una evolución hacia computadoras menos potentes para cada persona, con lo que se tenía una relación uno a uno entre la máquina y el ser humano. Sin embargo, en la actualidad se está hablando de una tercer oleada en la que una persona puede tener muchísimas computadoras de tamaño pequeño a su disposición. A esto se lo denomina “Tecnología Calma”, que es aquella que se encuentra a nuestro alrededor sin que nos afecte demasiado y que a la vez nos permita realizar procesos computacionales mientras estemos haciendo alguna otra cosa. [3]

Todo lo anterior introduce a lo que se denomina computación ubicua, que se trata de un modelo de computación en la que define a cada elemento como un ente vinculado con el entorno y que posee una capacidad de proceso que se adapta a las necesidades del usuario que se encuentra en el medio. Es un concepto similar al de computación distribuida pero “fuera del escritorio” y en donde los objetos tienen mayor capacidad de proceso para realizar tareas tanto por separado como de manera colaborativa. Esta visión se la atribuye a Mark Weiser, un investigador de Xerox, que ve a la tecnología como un medio para un fin y como algo que debería quedar en segundo plano para permitir al usuario concentrarse completamente en la tarea que está realizando. [4]

Para que todo esto sea posible, se deben tener en cuenta ciertos requisitos como definir interconexiones entre los elementos en cualquier lugar, seguir estándares de redes y dispositivos para su interoperabilidad (técnicas de descubrimiento, identificación, autoconfiguración, etc.), obtener sistemas confiables y seguros, y utilizar técnicas de inteligencia artificial para que los objetos puedan adaptarse y colaboren con los usuarios.

Cualquier entidad u objeto está ligada a un contexto, ya que no puede ser siempre representado sencillamente enumerando sus partes porque siempre tendremos información implícita que no será descripta. El concepto de “contexto” ha ido tomando forma desde la información de localización de un objeto hasta ampliarlo al estado del tiempo, temperaturas, época del año, estados de ánimo del usuario, orientación, grado de atención del mismo y otros objetos y personas alrededor de él. Según Anind K. Dey, “*el*

contexto es cualquier información que puede ser usada para caracterizar la situación de una entidad, que es una persona, lugar u objeto considerado relevante para la interacción entre el usuario y una aplicación, incluyéndolos como tal." [5]

En relación a la computación ubicua, el contexto se utiliza en referencia a aspectos del entorno que rodean a un sistema que relaciona a una persona con una o varias computadoras, y éste es relativo a los componentes específicos de dicho sistema y a puntos específicos de interacción.

Dependiendo de sus características, el contexto se puede agrupar de muchas maneras:

- **referida al origen:** de usuario, de computación, del entorno
- **según el grado de complejidad:** interno o externo, lógico o físico, de alto nivel o de bajo nivel

Y por otro lado se definen los distintos usos que se pueden dar del contexto [6]:

- **como una entrada adicional de datos**, donde el contexto se usa como entrada para que la aplicación le dé alguna utilidad.

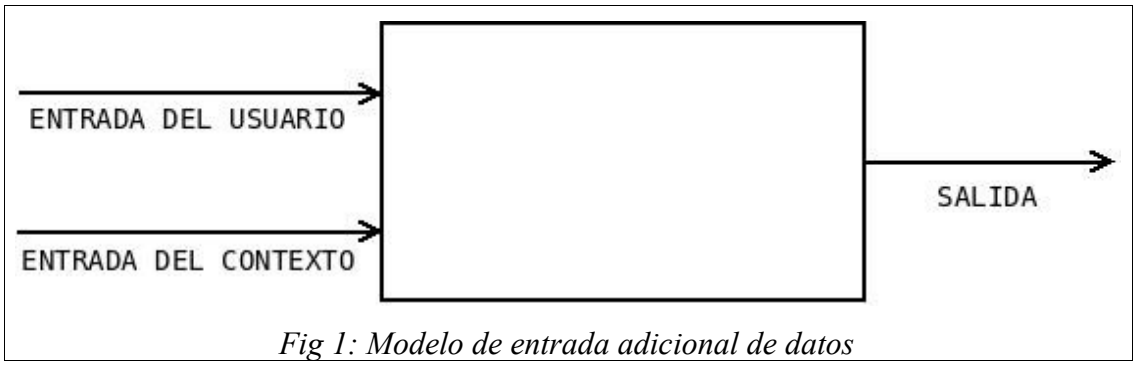


Fig 1: Modelo de entrada adicional de datos

- **como elemento que modifica la entrada de datos**, donde el contexto se utiliza como información para interpretar la entrada del usuario de manera correcta.

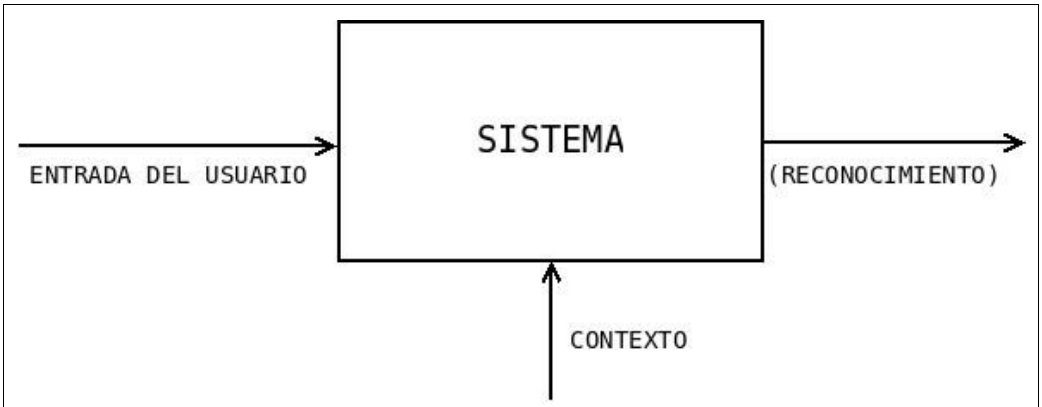


Fig 2: Modelo de elemento que modifica la entrada de datos

- **como información que vuelve al usuario**, en donde se crea un bucle en que el contexto es proporcionado de vuelta al usuario para que lo interprete y pueda modificar si lo desea su entrada de datos manual.

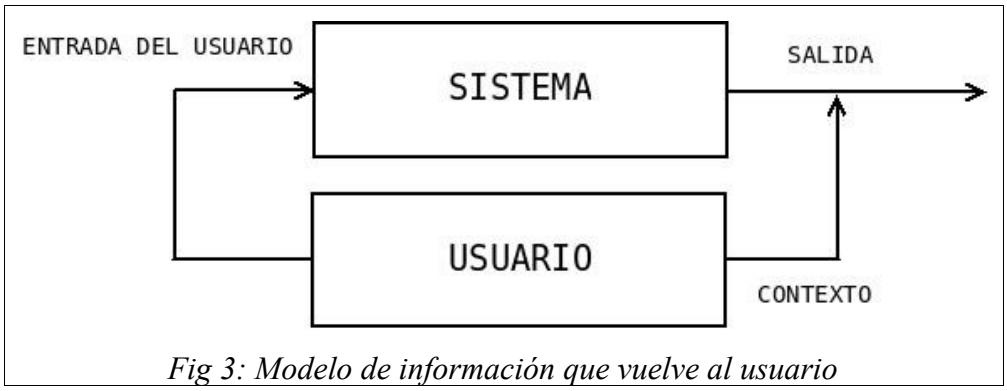


Fig 3: Modelo de información que vuelve al usuario

- **como un disparador de eventos**, en donde el sistema recibe la información de contexto y la analiza y dependiendo de ella puede disparar alguna serie de eventos.

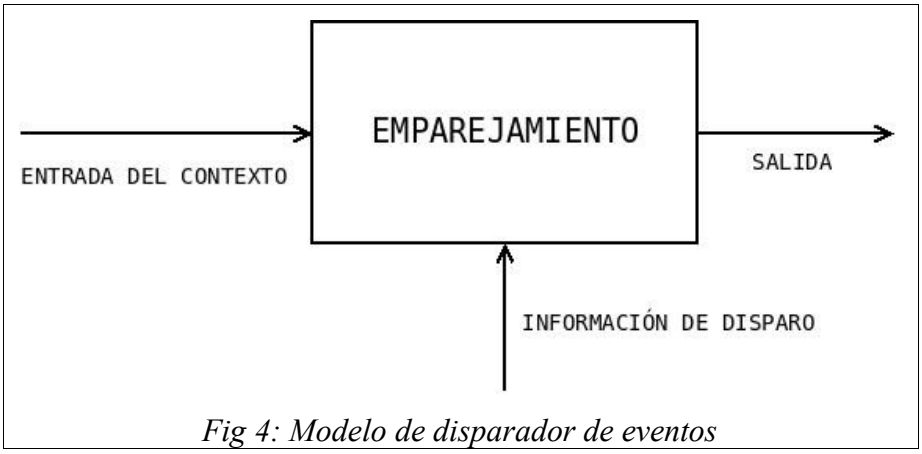


Fig 4: Modelo de disparador de eventos

Todo esto nos introduce a lo que se conocen como **aplicaciones sensibles al contexto**, que se pueden definir como sistemas capaces de extraer, interpretar y utilizar información del contexto para adaptar su comportamiento de acuerdo a ella. Esto se basa en varias tecnologías independientes; para la entrada de datos se usan sensores, que tienen sus propias limitaciones, para el procesamiento e interpretación de dichos datos depende de la tecnología y software especializado sobre inteligencia artificial, los cuales a su vez dependen de reglas y datos extraídos del entorno, que pueden incluir las preferencias del usuario. La clave de este tipo de aplicaciones es que se recopile experiencia y use dicho conocimiento para su objetivo.

Luego se dice que un sistema es sensible al contexto si posee la capacidad para proveer información relevante y/o servicios al usuario, que dependen de la tarea que este usuario está realizando. [7]

La sensibilidad al contexto (o *context-aware* en inglés) es una de las áreas principales dentro de la computación ubicua, que promete una interacción más fácil y natural entre el usuario y el sistema. Una limitación que tienen este tipo de aplicaciones es la carencia de un modelo que permita el desarrollo de las mismas tanto para el proceso de diseño como de implementación.

Este tipo de aplicaciones pueden categorizarse de diversas maneras que se generalizan en dos dimensiones: una referida a si las aplicaciones tienen como objetivo tomar o presentar información o ejecutar un comando, y otra referida a si las tareas son ejecutadas manual o automáticamente.

Dependiendo de esto, se pueden definir cuatro tipos principales de aplicaciones:

- **Selección por cercanía:**

Interfaz de usuario en donde los objetos que se encuentran alrededor son más fáciles de seleccionar que el resto. Ejemplo: una aplicación que mantenga información acerca de la localización del usuario, de manera tal que cuando éste cambia de lugar, la aplicación siempre intente reflejar en su interfaz los elementos más cercanos.

- **Reconfiguración contextual automática:**

Los componentes que se encuentran en el entorno se reconfiguran dependiendo del

contexto, es decir, agregar o quitar componentes, y eliminar o crear conexiones entre ellos. Ejemplo: configurar una sala de reuniones en donde se van creando componentes a medida que entran personas y se crean conexiones entre ellos. Además se puede crear una conexión entre cada persona y una pizarra.

- **Comandos contextuales:**

Aplicaciones que ejecutan comandos manuales del usuario pero que los mismos dependen del contexto. Ejemplo: cuando se quiere imprimir en la impresora más cercana al usuario. El comando es disparado manualmente y la búsqueda de la impresora se adapta a la locación.

- **Acciones originadas por el contexto:**

Aplicaciones en donde se ejecutan comandos automáticamente basándose en el contexto y en reglas o preferencias de usuario. Ejemplo: cuando una persona llega a una habitación y la luz se enciende de forma automática. [8]

Por otro lado, Anind K. Dey las clasifica según el uso que se le da al contexto:

- **Presentación de la información y los servicios al usuario**, donde el mismo recibe la información del contexto, que a la vez pueden ser servicios.

- **Ejecución automática de un servicio para el usuario**, que utilizan la información del contexto para disparar servicios dada una condición determinada y, a veces, respetando las preferencias del usuario.

- **Etiquetado del contexto de información para usos futuros**, que la utilizan para poder agregarla a contenido y datos que el usuario y el sistema genera o recibe.

Sin embargo, una aplicación puede ser clasificada bajo varias categorías a la vez, dado que puede tener un uso mixto de todo el contexto que la comprende.

1.5 Frameworks Context-Aware

Para crear aplicaciones sensibles al contexto (o context-aware) se emplean

distintas técnicas y aproximaciones. Los frameworks fueron ideados para ayudar al proceso de desarrollo de este tipo de aplicaciones y por ende intentar solucionar muchas problemáticas que se dan en estos entornos.

Una completa clasificación de los requisitos que debe tener una buena arquitectura *Context-Aware* la dio Mathias Baldauff [9], un investigador en un Centro de Competencias para la Investigación Industrial y Desarrollo de Austria, que propone las siguientes áreas como base de todo framework:

- **Sensorización:**

Abstracción y desacoplamiento de los detalles físicos de los sensores con relación a la lógica de la aplicación. Los sensores pueden no ser físicos.

- **Modelado de Contexto:**

Representación del contexto en forma de modelo para su posterior utilización, mantenimiento y actualización.

- **Procesamiento de Contexto:**

Razonamiento del contexto, lo que implica tranformarlo y utilizar su conocimiento.

- **Descubrimiento de Contexto:**

Mecanismos que descubren y localizan componentes dentro del contexto.

- **Datos Históricos de Contexto:**

Almacenamiento de la información de contexto junto con historial de los datos que se tomaron a lo largo del tiempo para un posterior análisis de su evolución.

- **Seguridad y Privacidad:**

Mecanismos de protección de los datos privados de un usuario o de aquellos que sean sensibles de alguna u otra forma.

Un framework para dispositivos móviles debe estar preparado para distintos tipos de cambios, como ser de contexto, de lugar, de conexión, de descubrimiento de nuevos sensores dentro de la misma locación o de desaparición de los mismos, etc. Procederemos a describir un framework conceptual que abarque los requisitos presentados.

Un **dispositivo de contexto** es un componente de software que provee aplicaciones con acceso a la información de contexto de su entorno. Ocultan las especificaciones de los dispositivos de entrada que se utilizan desde la aplicación y que permiten cambios con un mínimo impacto en ellas, un manejo de interacción para obtener resultados de las acciones del usuario y un reuso de bloques de construcción.

Para poder dirigir operaciones de contexto específicas, existen 4 categorías adicionales de componentes:

- **Intérpretes:**

Son los responsables de implementar la abstracción de interpretación, que se refiere al proceso de levantar el nivel de abstracción de una porción de contexto. Generalmente, un intérprete toma información de una o más fuentes de contexto y produce una nueva pieza de información de contexto.

- **Agregadores:**

Son los encargados de recolectar las piezas de información de contexto que están relacionadas lógicamente dentro de un repositorio en común.

- **Servicios:**

Son componentes dentro del framework que ejecutan acciones a favor de las aplicaciones, controlando o cambiando información del estado en el entorno.

- **Descubridores:**

Son los responsables de mantener un registro de qué capacidades existen en el framework, incluyendo dispositivos, intérpretes, agregadores y servicios que están actualmente disponibles para el uso de las aplicaciones. Cuando cualquiera de estos componentes fue iniciado, se notifica junto con sus capacidades y su interfaz de uso (lenguaje, protocolo, etc.). Cuando alguno de ellos falla, es la responsabilidad de un descubridor determinar que el componente no se encuentra más disponible para su uso.

Por lo tanto, los dispositivos indican qué tipo de contexto proveen, los intérpretes qué interpretaciones pueden realizar, los agregadores qué entidad representan y el tipo de contexto que pueden proveer de ella, y los servicios indican qué servicio sensible al contexto puede proveer y el tipo de contexto e información requerida para su ejecución.

Veamos a un ejemplo de configuración de estas componentes [10]:

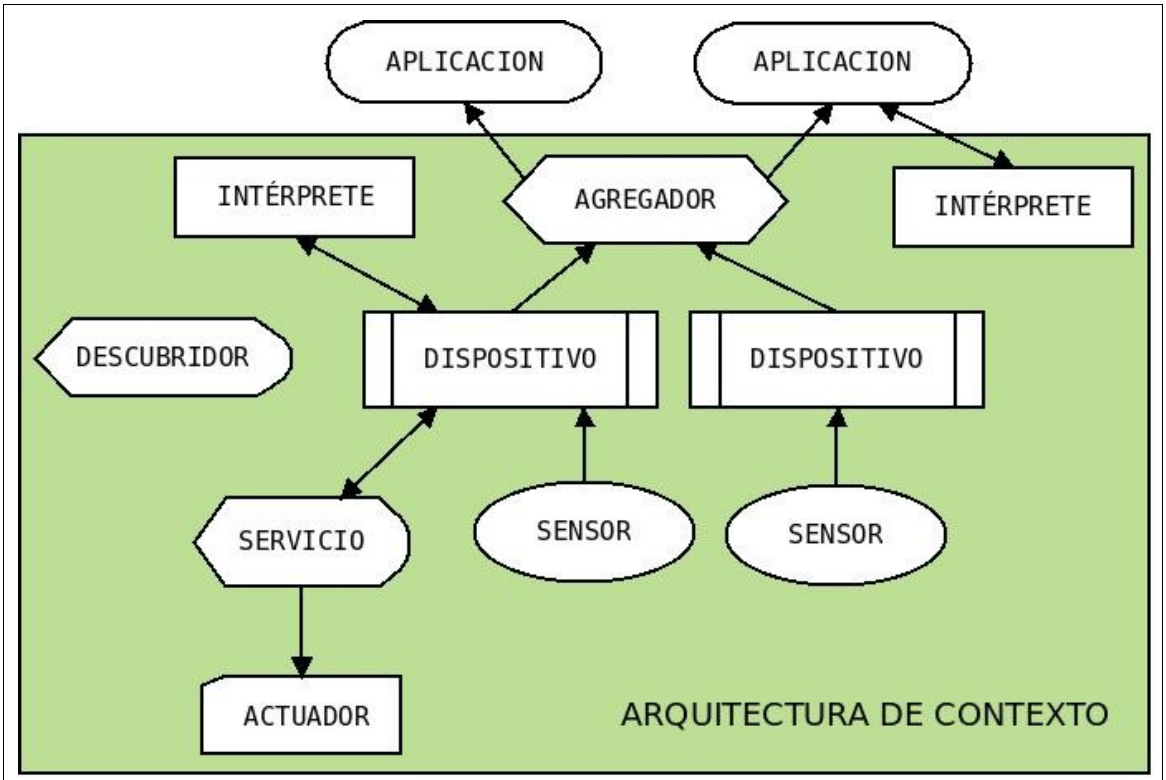


Fig 5: Configuración de ejemplo de las componentes en herramientas de contexto

Se disponen de dos sensores, dos dispositivos, un agregador, dos intérpetes, un servicio, un descubridor y dos aplicaciones. Cuando una componente de contexto está disponible, se registran sus capacidades con un descubridor, lo que le pemite a los agregadores encontrar dispositivos relevantes e intérpretes y le permite a las aplicaciones encontrar agregadores relevantes, dispositivos e intérpretes. Un sensor le provee información al dispositivo de contexto, el cual la almacena, puede llamar a un intérprete para obtener un nivel de abstracción más alto de los datos y luego le habilita al contexto a otras componentes y aplicaciones. Un agregador toma el contexto de los dispositivos que pueden proveer contexto de la entidad que representa, y también guarda el contexto recibido, puede llamar a un intérprete y luego le habilita el contexto al resto. Por último, las aplicaciones pueden consultarle a los agregadores y pueden llamar a los intérpretes en el

caso en que el nivel de abstracción deseado no está disponible para los dispositivos y los agregadores.

Todas estas componentes son ejecutadas independientemente de las aplicaciones, lo que permite la obtención constante del contexto y el uso por varias aplicaciones. Además, todas las componentes y aplicaciones se comunican entre sí automáticamente por medio de lenguajes y protocolos red conocidos, lo que posibilita al programador que implemente un componente en particular o una aplicación para comunicarse con otras componentes sin tener que preocuparse de los mecanismos usados para la comunicación.

2 Tecnología Bluetooth

La palabra **Bluetooth** proviene de un rey danés y noruego del siglo X llamado Harald Blåtand (traducido como Harold Bluetooth) que logró unir a las tribus danesas, noruegas y suecas en un solo reino. De ahí la implicancia de utilizar dicha palabra para definir esta tecnología, la cual unifica los protocolos de comunicación en un estándar universal. Luego, se define Bluetooth como una tecnología de ondas de radio de corto alcance cuyo objetivo es simplificar los enlaces entre dispositivos electrónicos.

Si bien hay muchas otras maneras de conectar dispositivos (cables, señales de radio, rayos de luz infrarrojos, etc.), el Bluetooth es una tecnología automática e inalámbrica con características muy interesantes que pueden facilitar muchas cosas.

En 1998 se fundó lo que se denomina el Bluetooth SIG (Special Interest Group), que es una asociación privada sin fines de lucro integrada por miles de compañías de telecomunicaciones, informáticas, de tecnologías de red y otros rubros con el objetivo de ofrecer soporte para Bluetooth tanto para su desarrollo, implementación y comercialización en los productos que la proveen.

2.1 Clases y Frecuencias

Bluetooth opera en el rango de frecuencia 2,4 a 2,48 Ghz con amplio espectro, que generalmente está disponible en la mayor parte del mundo y es de uso libre y abierto a cualquier sistema de radio. Habitualmente, su alcance es menor a los 10 metros; sin embargo, se pueden alcanzar distancias de hasta 100 metros mediante el uso de amplificadores. Según su potencia, los dispositivos Bluetooth se pueden clasificar en tres clases:

Clase	*Potencia Máxima permitida*	*Rango aproximado*
1	*100 mW (20 dBm)*	*100 m*
2	*2,5 mW (4 dBm)*	*10 m*
3	*1 mW (0 dBm)*	*1 m*

Tabla 1: Potencia y rango por clase de sensores

Los dispositivos de clases 2 y 3 pueden implementar opcionalmente un control de potencia alternativo, que siendo necesario para los de clase 1, este mecanismo permite que un radio Bluetooth reduzca su potencia al mínimo nivel necesario para mantener su enlace, y de esta manera ahorrar energía y reducir las posibilidades de interferir con otras redes cercanas. De todas maneras estas distancias son relativas, ya que se miden punto a punto y a campo abierto. En la práctica, en instalaciones normales dentro de un edificio, dichas distancias oscilan entre 5 y 25 metros. Pero existen ciertos dispositivos en los que la señal se puede amplificar hasta un nivel por encima del máximo permitido por la tecnología, como por ejemplo adaptadores USB cuyo alcance supera los 150 metros. Por otro lado, mediante diversas técnicas y antenas más potentes se pueden obtener alcances de entre 1 o 2 km.

La mayoría de los teléfonos celulares, auriculares, laptops y otros dispositivos Bluetooth de nivel consumidor son de clase 2. Los de clase 1 también están disponibles al consumidor, pero los de clase 3 ya casi no existen.

Los saltos de frecuencia posibles para transmitir en full duplex (ida y vuelta simultáneamente) son un máximo de 1600, y se dan entre un total de 79 frecuencias con intervalos de 1Mhz, lo que permite dar seguridad y robustez. Esto los diferencia de otras tecnologías, ya que divide el ancho de banda en los 79 canales disponibles y emplea una técnica aleatoria para que los dispositivos estén cambiando permanentemente la frecuencia por la cual envían y reciben los datos.

2.2 Especificaciones

La documentación de Bluetooth está dividida en dos secciones: especificación y perfiles. La especificación describe cómo trabaja la tecnología y los perfiles cómo es

usada la misma.

La especificaciones principales (denominadas *core*) contienen la pila de protocolos Bluetooth, la cual permite localizar, conectar e intercambiar datos entre dispositivos y ejecutar aplicaciones interactivas e interoperables entre sí.

La pila está compuesta por varias capas que, basadas en el modelo OSI[2], se graficarían de la siguiente manera:

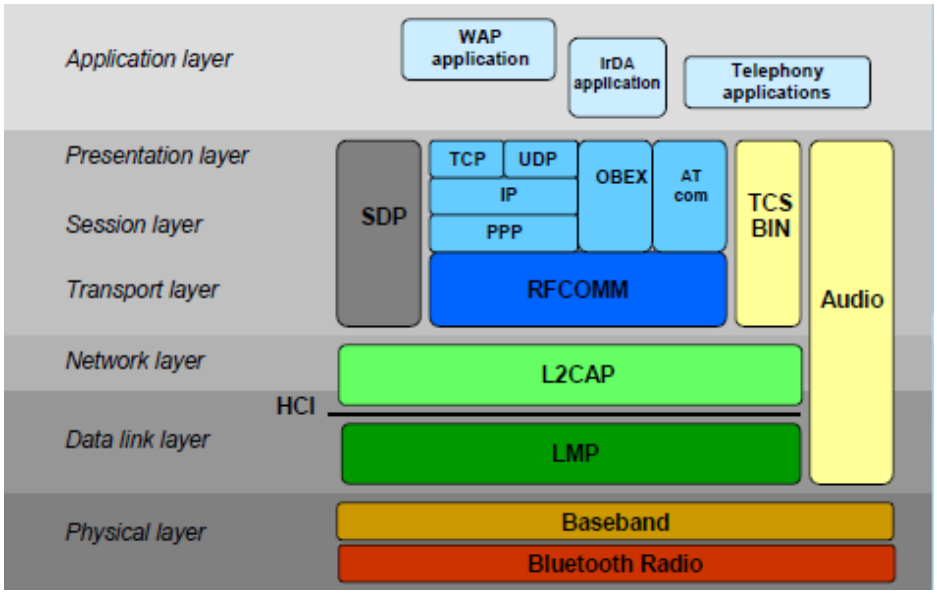


Fig 6: Pila de protocolos Bluetooth

2.2.1 Radio Bluetooth:

Es la capa más baja de la pila y define los requisitos del transceptor Bluetooth (dispositivo que envía y recibe) en la banda de radiofrecuencia utilizada por la tecnología. Implementa la comunicación física y especifica detalles de la interfaz de aire como ser las bandas y saltos de frecuencia, el arreglo de canales, el esquema de modulación y los niveles de potencia permitidos de transmisión.

2 Marco de referencia para definir arquitecturas de interconexión entre distintos dispositivos

2.2.2 Banda Base:

Se encuentra por encima de la capa de Radio y se encarga de manejar los canales y enlaces físicos, además de otros servicios tales como corrección de errores, selección de saltos, seguridad, etc. Esta capa es implementada como un Controlador de Enlace, el cual trabaja con el manejador de enlaces para llevar a cabo rutinas de nivel de enlace como conexiones y control de potencia. También gestiona enlaces sincrónicos y asincrónicos, maneja paquetes y realiza paginación y descubrimiento para acceder y examinar dispositivos Bluetooth dentro del área. El transceptor aplica un esquema de división del tiempo duplex y además el tiempo también se divide para diferentes saltos de frecuencia.

2.2.3 LMP (Link Manager Protocol):

Esta capa es la responsable de establecer el enlace entre dispositivos Bluetooth y administrar el enlace establecido. Especifica y gestiona el establecimiento y cierre de la conexión mediante el intercambio de mensajes de control, gestiona el consumo de energía, el tipo del enlace (sincrónico o asincrónico), y realiza autenticación y encriptación de los paquetes transmitidos para proveer seguridad.

2.2.4 HCI (Host Controller Interface):

Se trata de una interfaz entre el host y el controlador que se encarga de los mecanismos de transporte de datos, una vez que el enlace fue establecido, y de segmentar los datos si la aplicación lo requiere. La misma contiene una interfaz de comando para el controlador de banda base y la gestión de enlace y para acceder al estado del hardware y sus registros.

2.2.5 L2CAP (Logical Link Control and Adaption Protocol):

Esta capa está por encima del HCI y provee servicios orientados a conexión y sin conexión a los protocolos de capas superiores con la capacidad de segmentación y

reensamblado de los datos y abstacciones de grupos. Este protocolo le permite a la aplicaciones y a los protocolos de nivel más alto transmitir y recibir paquetes de datos de hasta 64 kilobytes de longitud.

2.2.6 RFCOMM (Radio Frequency Communications):

Este protocolo emula los puertos serie sobre el protocolo L2CAP. Lo que hace es emular señales de control y datos sobre la banda base Bluetooth, ofreciendo capacidades de transporte a servicios de capas superiores que usan una línea serial como mecanismo de transporte. Está diseñado para que el reemplazo de tecnologías de cable sea tan transparente como sea posible.

2.2.7 SDP (Service Discovery Protocol):

Este protocolo proporciona un medio para que las aplicaciones descubran los servicios disponibles y determinen sus características. Esto es necesario de especificar en un ambiente Bluetooth debido a que el conjunto de servicios disponibles cambian dinámicamente ya que se basan en la proximidad por radio frecuencia de los dispositivos. Esta capa sirve como interfaz para aplicaciones como las videoconferencias, y también para actuar sobre dispositivos como impresoras o faxes.

La capa de **Audio** es una capa especial utilizada solamente para el envío de audio sobre Buetooth, cuyas transmisiones pueden ser realizadas entre una o más unidades usando varios modelos diferentes. Los datos de audio no pasan a través de la capa L2CAP pero sí cuando abre un enlace y cuando establece de manera directa dos unidades Bluetooth.

El **Control de Telefonía Binario (TCS BIN)** es un protocolo que define la señalización de control de llamadas para establecer y liberar una conversación o una llamada de datos entre unidades Bluetooth. También ofrece funcionalidad de intercambio de información de señalización que no esté relacionada con el progreso de llamadas.

Por encima de RFCOMM tenemos varios protocolos específicos ya publicados por otras organizaciones que detallaremos a continuación:

- **AT com:**

  Son comandos para el control de telefonía a través de emulación de puerto serial

- **PPP (Point-to-Point Protocol):**

  Es un protocolo que transmite datagramas IP (fragmentos de paquetes) mediante un enlace punto a punto, por lo que utiliza su mecanismo serial para conviertir el fujo de los paquetes.

- **TCP/UDP – IP:**

  Estos estándares le permiten a los dipositivos Bluetooth conectarse, por ejemplo, a Intenet a través de otros dispositivos conectados, por lo que puede actuar como un puente para la conexión.

- **OBEX (Object Exchange Protocol):**

  Es un protocolo desarrollado por IrDA (Infrared Data Association) para el intercambio de objetos, que provee funcionalidad similar al HTTP pero de una manera más sencilla. También ofrece un modelo para representar objetos y operaciones. Utiliza un modelo cliente-servidor y es independiente del mecanismo y la de la API (Application Program Interface) de transporte.

Por último se encuentra el **WAP (Wireless Aplicatition Protocol)** que es una especificación de protocolo inalámbrica que trabaja con varias tecnologías de red inalámbricas conectando dispositivos móviles a Internet. Luego, Bluetooth puede ser utilizado como portador para transportar los datos entre el cliente WAP y su servidor WAP adyacente, lo que le permite brindar al cliente posibilidades únicas en cuanto a la movilidad comparado con otros portadores WAP.

2.3 Perfiles

Para que todos los dispositivos que usen Bluetooth sean compatibles entre sí

existen los perfiles, los cuales definen una descripción de la interfaz de comunicación entre dos unidades para un servicio particular. Determinan las aplicaciones posibles y especifican el comportamiento general que un dispositivo Bluetooth utiliza para comunicarse con otro.

Hay una amplia gama de perfiles que describen muchos tipos de aplicaciones y casos de uso para dispositivos. Mínimamente cada perfil debe contener información acerca de dependencias de otros perfiles, formatos de interfaz de usuario sugeridos y partes específicas de la pila de protocolos usada por el mismo. Cada perfil nuevo puede ser construido sobre alguno ya existente, lo que permite reusar eficientemente los protocolos y procedimientos existentes.

A continuación se verá un gráfico y detalles de los perfiles más importantes:

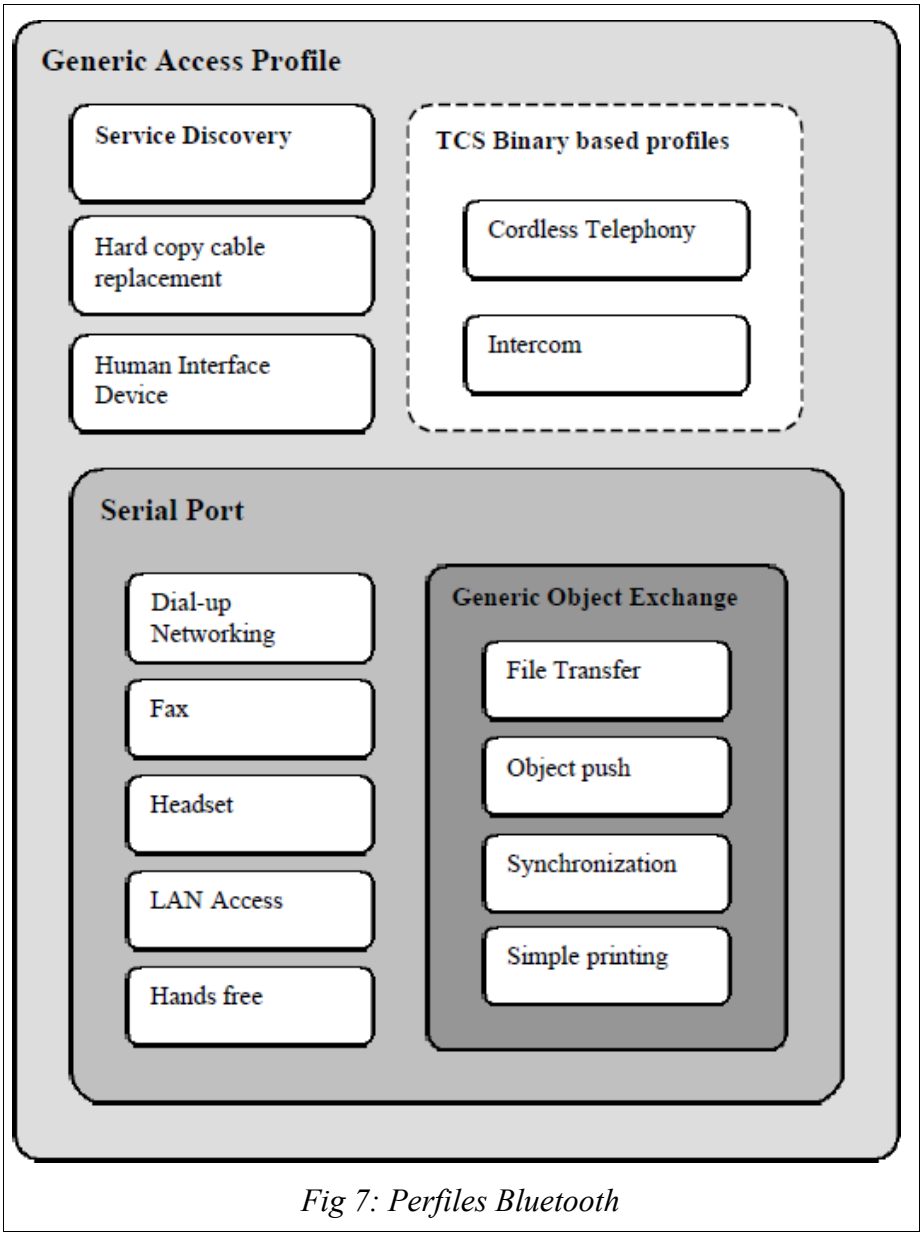


Fig 7: Perfiles Bluetooth

2.3.1 General Access Profile (GAP):

Este perfil es necesario para todos los modelos de uso y define la forma en que se descubren y se conectan los dispositivos Bluetooth, así como también define protocolos de seguridad. Todos los dispositivos deben ajustarse por lo menos a él para asegurar la interoperabilidad básica.

Service Discovery:

Conocido como SDAP (Service Discovery Application Profile), utiliza partes del GAP para definir el descubrimiento de servicios para los dispositivos.

2.3.2 Serial Port:

Define cómo setear y conectar puertos seriales virtuales entre dos dispositivos. Esta emulación puede ser usada para tareas tales como transferencia de datos e impresiones.

2.3.3 Generic Object Exchange (GOEP):

Depende del perfil de Serial Port y es usado por las aplicaciones para gestionar el intercambio de objetos, que luego es usado por otros perfiles para realizar funciones como carga de objetos, transferencia de archivos y sincronización.

Object Push:

Este perfil es utilizado para intercambiar objetos pequeños tales como imágenes, documentos, sonidos, contactos (vCards) o eventos (vCalendar).

File Transfer:

Se utiliza para transferir archivos entre dos dispositivos Bluetooth y se diferencia del anterior perfil ya que en este caso un dispositivo puede acceder al sistema de archivos

del otro para enviar o recibir los archivos.

Sincronización:

Este perfil permite sincronizar calendarios e información de contactos entre dos dispositivos.

2.4 Redes Bluetooth: Piconets y Scatternets

Dentro de una red de dispositivos, un dispositivo Bluetooth puede ser tanto maestro (nodo central) como esclavo (resto de los nodos), dependiendo del escenario de la aplicación. La tecnología emplea para la comunicación saltos de frecuencia de espectro ampliado (FHSS), por lo que luego, para que varios dispositivos puedan comunicarse, deben sincronizarse en la misma secuencia de salto. El maestro setea el salto de frecuencia y los esclavos sincronizan con él.

Una **piconet** está formada por un maestro (*master*) y puede tener hasta siete esclavos (*slaves*), los cuales sólo se pueden comunicar con el maestro, como se ve en la siguiente figura:

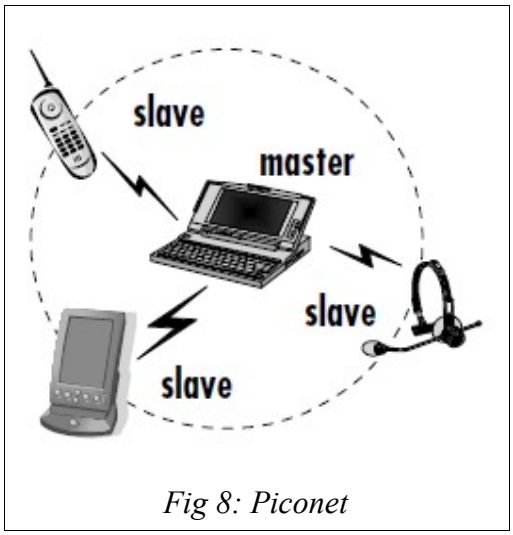


Fig 8: Piconet

En cambio una **scatternet** está formada al relacionar dos o más piconets. Cuando un dispositivo se encuentra en más de una piconet, debe usar tiempo compartido y sincronizarse con el maestro de la piconet con la que se esté comunicando. Por lo tanto un dispositivo en una scatternet puede ser maestro en una piconet y esclavo en otra, como se ve a continuación:

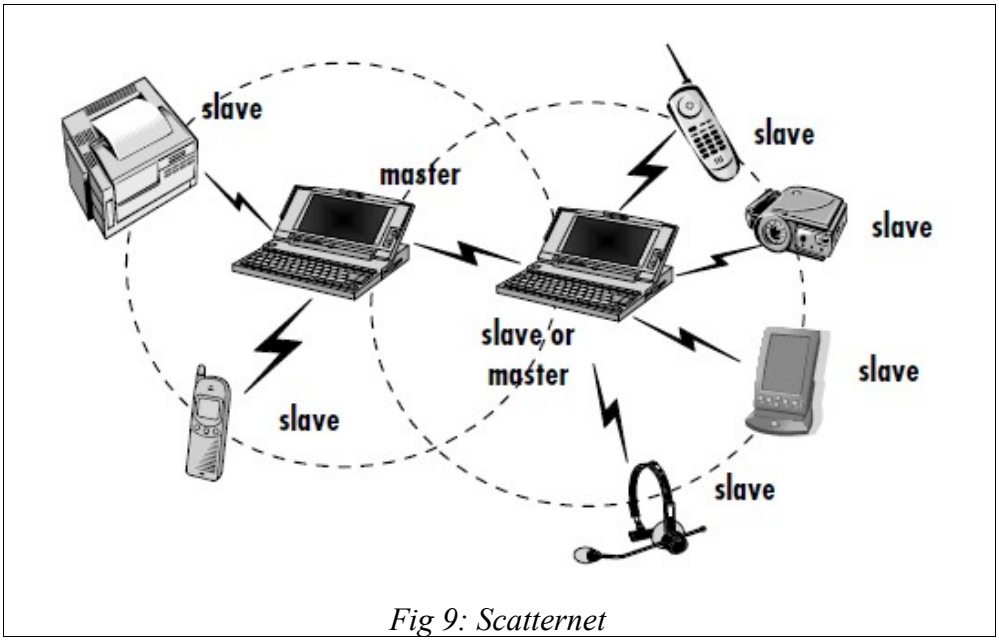


Fig 9: Scatternet

Todos los dispositivos tienen la capacidad de tomar tanto el rol de esclavo como el de maestro, por lo que los esclavos pueden decidir de crear otra piconet. Este concepto es importante para el rendimiento agregado, ya que muchas piconets pueden funcionar de forma paralela. Luego, al poder ser un dispositivo miembro de dos piconets, es posible intercambiar información entre ambas por medio del nodo en común, aunque hay problemas prácticos con esto respecto a los tiempos de transferencia y a la calidad del servicio cuando un dispositivo se encuentra ausente de una piconet. Una limitación es que un dispositivo puede ser solo el maestro en a lo sumo una de las piconets a las que pertenece.

2.5 Comunicación entre los dispositivos

Cada chip de Bluetooth posee una dirección global única de 48 bits, que se denomina BD_ADDR, y que la gestiona la Autoridad de Registro IEEE. De esta manera un dispositivo se puede identificar de otros dispositivos, y si el mismo es maestro, el tiempo de conexión y la secuencia de salto también son derivados de dicha dirección. Pero las personas no se manejan con estas direcciones de números, sino que utilizan un nombre “amigable” que se puede instanciar en cada dispositivo que son similares a los DNS debido a que son nombres fáciles de recordar que se traducen en identificadores que entienden las máquinas. Pero ambos difieren en que los nombres de dominio son únicos y

registrados en una base de datos centralizada, mientras que los nombres de los dispositivos Bluetooth son arbitrarios, se pueden duplicar y sólo se registran en el dispositivo de forma local.

Para buscar un dispositivo, en TCP/IP se traduce el nombre de dominio en una dirección IP contactándose con un servidor de nombres, realizando una consulta y esperando un resultado. En cambio con Bluetooth, el proceso de búsqueda es al revés; primero un dispositivo busca dispositivos cercanos a él y recopila sus direcciones para luego contactarse con cada uno de manera individual y preguntándoles por su nombre. [11]

El proceso de búsqueda de dispositivos Bluetooth cercanos consiste en difundir un mensaje de “descubrimiento” en broadcast y esperar respuestas, donde cada una de ellas contiene la dirección BD_ADDR y un entero que identifica la clase del dispositivo (por ejemplo si es un celular, una computadora, unos auriculares, etc.), y el resto de los datos del dispositivo se pueden obtener luego contactando a cada uno en particular. La mayor contra de este proceso es el tiempo de detección de todos los dispositivos cercanos sumado al tiempo adicional en que se detecta un nuevo dispositivo que ha sido ubicado recientemente en las cercanías del dispositivo que está realizando la búsqueda, lo que da un promedio total de 10 a 15 segundos. Esta demora se debe a que durante el proceso de búsqueda, el celular está cambiando de frecuencia más de mil veces por segundo y existen solamente 79 frecuencias posibles en las cuales se puede transmitir. Versiones más recientes de Bluetooth (como la 1.2) intentan reducir este tiempo, pero no lo hacen de una manera considerable. No hay manera que un nuevo dispositivo anuncie su presencia; el mismo debe ser encontrado por un proceso que lo busque.

Existen dos alternativas para determinar si un dispositivo responde o no al proceso de descubrimiento y conexión: el escaneo de petición y el escaneo de página. Dependiendo de la combinación de ambas, se lee una interpretación diferente:

Escaneo de Petición	*Escaneo de Página*	*Interpretación*
ENCENDIDO	*ENCENDIDO*	*El dispositivo es detectable por otros y acepta solicitudes de conexión entrantes.*
APAGADO	*ENCENDIDO*	*A pesar de no ser detectable por otros, el dispositivo aún responde a solicitudes de conexión de dispositivos que ya tienen su BD_ADDR .*
ENCENDIDO	*APAGADO*	*El dispositivo es detectable por otros pero no acepta solicitudes de conexión entrantes.*
APAGADO	*APAGADO*	*El dispositivo no es detectable por otros ni acepta solicitudes de conexión entrantes.*

Tabla 2: Combinaciones posibles para el descubrimiento y conexión de un dispositivo Bluetooth

Generalmente se encuentran dispositivos seteados por defecto con ambas opciones encendidas. La tercer combinación no es muy frecuente y la última se utiliza para aquellos dispositivos que sólo establecen conexiones de salida.

Si ya se realizó el proceso de búsqueda, no es necesario repetirlo cada vez que se reestablece un enlace entre dos dispositivos, ya que utiliza energía de manera innecesaria y permite a otro dispositivo ser encontrado dentro del radio, lo que genera respuestas a peticiones innecesarias. El dispositivo puede además intentar conectarse una vez más para verificar el nombre del dispositivo, con lo que gastaría aún más energía. Es por eso que muchos dispositivos se encuentran sólo con el Escaneo de Página encendido, como en el caso de los auriculares, que son emparejados previamente. Cuando un dispositivo que ya ha sido emparejado se enciende, puede estar en modo de Escaneo de Página y el otro lo comunica.

2.6 Búsqueda de Información de Servicios

Existen diferentes tipos de dispositivos Bluetooth, cada uno con diferentes combinaciones de perfiles posibles; pueden conectarse y hablar entre sí pero pueden no soportar perfiles compatibles. Se pueden consultar todos los servicios disponibles o si soporta un servicio en particular. El mecanismo de solicitud y respuesta de esta consulta

lo provee el protocolo SDP, que accede a la base de datos de servicios que brinda el dispositivo, y la cual también contiene información necesaria para saber cómo utilizar cada uno. Luego se guarda un registro que contiene **atributos** que describen distintos tipos de información, como ser el servicio que brinda el dispositivo (nombre, disponibilidad, descripciones), los protocolos utilizados para acceder a los servicios ofrecidos (L2CAP, RFCOMM), cómo conectarse a dichos protocolos (puertos), los perfiles soportados, cómo se construye el árbol de exploración de servicios y el comportamiento de la base de datos (como por ejemplo cuando un registro de servicio es actualizado). [12]

Los atributos se identifican por su propia **UUID** (*Universally Unique Identifier*), que es un identificador numérico de 128 bits, generado de tal manera que garantiza su unicidad en tiempo y espacio e identifica a un servicio dado. También hay UUIDs más cortas, de 16 o 32 bits. Expresamente, un desarrollador elige su UUID en tiempo de diseño y cuando el programa es ejecutado, graba un registro de servicio que contiene el ID del servicio con el servidor SDP para el dispositivo. Una aplicación cliente que busque un servicio particular consultará el servidor SDP de cada dispositivo dentro de su búsqueda si el dispositivo ofrece algún servicio con el mismo UUID.

Por otro lado, dentro del registro del servicio se tiene otro identificador llamado **ID de Clase de Servicio**, que permite clasificar a los servicios. Es decir, que si dos compañías diferentes distribuyen dos aplicaciones Bluetooth distintas pero que hacen lo mismo, deberían tener el mismo ID de Clase de Servicio (previo acuerdo entre ambas compañías). Pero si una aplicación provee más de un servicio, se pueden tener dos soluciones: se pueden registrar dos servicios por separado, cada uno con un ID de clase de servicio específico, o bien, como se da en la mayoría de los casos, se tiene cada servicio asociado a una lista de ID de clases de los servicios que provee.

Atributos SDP comunes:

Bluetooth define varios atributos reservados que tienen un significado particular, y el resto puede ser usado arbitrariamente. Algunos de los atributos más comunes son:

- **Lista de IDs de Clases de Servicio:** Una lista de UUIDs de clases que provee el servicio (es obligatorio que esté en el registro).
- **ID de Servicio:** Un UUID simple que identifica un servicio específico.
- **Nombre del Servicio:** Una cadena de texto que describe el nombre del servicio.
- **Descripción del Servicio:** Una cadena de texto que describe el servicio provisto.
- **Lista de Descriptor de Protocolo:** Una lista de protocolos y números de puerto

usados por el servicio.

- **Lista de Descriptor de Perfil:** Una lista del descriptor de perfil Bluetooth con la que cumple el servicio. Cada descriptor consiste en una UUID y un número de versión.
- **Manejador de Registro de Servicio:** Un número entero sin signo que identifica unívocamente al registro dentro del dispositivo.

2.7 Comunicación vía Sockets

Un *socket* en programación de redes representa el punto final de un enlace de comunicación. La idea es que desde el punto de vista de una aplicación, toda la información enviada a través del enlace debe entrar o salir del socket. Existen dos maneras para usarlo: como cliente o como servidor, dependiendo si se crean conexiones salientes o si se aceptan conexiones entrantes.

Una vez que una aplicación Bluetooth tiene un socket conectado, usarlo para comunicaciones es muy sencillo. Se utilizan los comandos *send* y *receive* para enviar y recibir datos. Se envían los bytes, pero su respuesta satisfactoria solamente significa que los bytes han sido movidos del espacio de direcciones de la aplicación a los buffers del sistema operativo; no necesariamente implica que hayan liberado el dispositivo. La recepción devuelve al menos algunos datos, salvo que la conexión se haya roto. Cuando la aplicación finaliza, invoca al comando *close* para desconectar el socket. Al cerrar un socket del servidor, se desconecta del puerto y no acepta más conexiones entrantes.

En los procesos de comunicación existe una cierta espera, en donde el *thread* (hilo de ejecución) que controla el enlace bloquea al resto y los deja sin poder hacer nada, como por ejemplo, responder al usuario o trabajar en otro socket. Esto puede ser positivo para muchas aplicaciones sencillas, pero no es aceptable para aplicaciones más complejas que necesiten realizar otras tareas de manera simultánea. Para evitar esto, se pueden utilizar múltiples *threads* de control, con un *thread* dedicado a cada tarea que requiera algún bloqueo, pero al ser una solución bastante complicada, se utilizan otras técnicas asincrónicas. La herramienta principal en la programación de sockets asincrónicos es un **bucle de eventos simple**, cuya idea es que la aplicación espere por todo de una sola vez, y no esperar por un solo evento, como puede ser una conexión entrante o nuevos datos recibidos. Cuando ocurre un evento esperado, la aplicación lo procesa y retorna al ciclo de eventos a la espera de más sucesos. Los eventos deben ser procesados de forma rápida y los llamados a funciones no deben bloquear nada y deben retornar sin demasiada demora. Para ello se configuran los sockets en modo “no

bloqueante", ya que por defecto un socket Bluetooth es "bloqueante". Como dichas operaciones "no bloqueantes" no avisan a la aplicación cuándo sucede cada evento, es necesario disponer de una manera diferente para esperarlos. Para ello se utiliza la operación **select**, la cual espera a que suceda algún evento o a que se termine un tiempo de expiración.

2.8 Limitaciones

A pesar de que Bluetooth es una gran tecnología inalámbrica, no tiene las mismas características que el resto; algunas de ellas ni siquiera están presentes o bien se encuentran limitadas. Se enumerarán algunas de ellas:

- Anunciar la presencia de un dispositivo.
  Si bien es posible buscar dispositivos cercanos, no es posible notificar al resto cuando uno se encuentra en el radio de búsqueda.

- Detectar cuando un dispositivo remoto está buscando dispositivos cercanos.
  A pesar de que el Bluetooth exhibe muchos eventos de bajo nivel, no lo hace con eventos que se disparan cuando se detecta un mensaje de búsqueda. El adaptador Bluetooth puede detectarlo, pero no comparte dicha información con la máquina. Se podría hacer una solución de hardware para la detección, pero dicha opción no es muy factible para muchos desarrolladores de software.

- Determinar la dirección Bluetooth de un dispositivo de búsqueda.
  Un dispositivo Bluetooth que lleva a cabo una búsqueda de otros dispositivos, nunca transmite información que se identifique a sí mismo. Por lo tanto, aunque fuera posible detectar desde un programa cuándo un dispositivo remoto se encuentra realizado una búsqueda, no sería posible identificarlo.

- Conocer la distancia a un dispositivo remoto.
  Aunque la señal de un dispositivo pueda ser más o menos fuerte dependiendo de la distancia del dispositivo que lo encontró, no sería un dato preciso debido a que el radiotransmisor se encuentra ajustando continuamente la fuerza de la señal para

evitar errores de transmisión y para conservar la potencia. Además la medida es muy relativa ya que pueden existir interferencias con otras radiotransmisoras o pueden atenuearse la señal por distintos materiales que se encuentren alrededor.

- Enviar mensajes en *broadcast*.

  No es posible que un dispositivo Bluetooth envíe un mensaje en *broadcast* (multidifusión) a todos sus dispositivos cercanos. Hay un protocolo para el maestro de una piconet para enviar un mensaje en broadcast a todos sus esclavos utilizando paquetes L2CAP no orientados a conexión, pero no es muy útil debido a que todos los dispositivos deberían estar en la misma piconet y además el protocolo no es confiable (no existe un acuse de recibo ni retransmisiones).

2.9 Seguridad

La información transmitida por Bluetooth puede ser sensible y vulnerable a escuchas. Los usuarios de celulares o laptops quisieran estar seguros de que no se les conecten personas no autorizadas por ellos. También existe la privacidad por locación, que implica que cualquier persona pueda utilizar sus dispositivos Bluetooth en cualquier lugar sin estar focalizado en que cualquiera pueda rastrear sus movimientos. Por eso es importante el anonimato del dispositivo utilizado.

Se esperan cuatro puntos fundamentales respecto a la seguridad:

1. Configuración de seguridad fácil de usar y autoexplicativa
2. Protección de confidencialidad
3. Autenticación de dispositivos conectados
4. Anonimato

La tecnología Bluetooth provee encriptación del enlace y autenticación. Para proveer anonimato existen varias soluciones posibles, pero la configuración de seguridad no es tan sencilla. [13]

Cuando se consideran sistemas de información en general, la seguridad debería abarcar los siguientes aspectos: confidencialidad, integridad y disponibilidad. Los mecanismos que se refieren a los aspectos de confidencialidad deberían proveer los medios para mantener privada la información del usuario; los referidos a la integridad la capacidad de proteger los datos contra modificaciones o borrados no autorizados; y la

disponibilidad se refiere a que el sistema debería estar tan disponible como se espera, por lo que dicho punto está fuertemente relacionado con la confiabilidad y la robustez. El estándar Bluetooth no incluye ningún mecanismo de protección de la integridad de los datos.

La seguridad básica provista por Bluetooth está basada en el uso de *mecanismos criptográficos de claves simétricas*[3] para la autenticación, encriptación de enlace y generación de claves, y dichos mecanismos utilizan distintos tipos de claves. Para verificar el establecimiento de un enlace entre dos dispositivos Bluetooth, se emplea un proceso de autenticación entre ambos, que utiliza la clave de enlace, la cual también se usa para derivar la clave de encriptación.

En una comunicación entre dos dispositivos se tienen dos estados respecto al *emparejamiento*[4]: cuando no están emparejados y cuando ya lo están. Esta operación de emparejarlos va a producir una clave de enlace entre ambos para autenticarlos y la generación de claves de encriptación directamente después de haberlo emparejado y en instancias posteriores. El sistema Bluetooth reconoce dos tipos de claves de enlace: *semipermanentes* y *temporales*, y a la vez se tienen dos subtipos claves semipermanentes: *claves de unidad* y *claves combinadas*. Una clave de unidad es aquella que es generada por una unidad en sí misma y usa como clave de enlace con cualquier otro dispositivo, y una clave combinada es aquella que genera un dispositivo en cooperación con otro. Luego, una clave combinada es más segura ya que la misma es conocida solamente por el mismo dispositivo y por el que le ayudó a generarla, mientras que la clave de unidad puede ser conocida por muchos otros dispositivos y es segura únicamente cuando se tiene confianza con los dispositivos con los que se emparejó usando la misma clave. Por otro lado, las claves temporarias también se dividen en dos subtipos: *claves de inicialización* y *maestra*. La clave de inicialización es una clave de corta duración y que existe solamente durante el emparejamiento de dos dispositivos. La clave maestra es una clave de enlace que el maestro genera antes de la configuración de una comunicación *broadcast* encriptada a varios dispositivos esclavos.

Un dispositivo Bluetooth en un estado “conectable” acepta solicitudes de conexión de otros dispositivos, lo que implica un riesgo de ataque por un dispositivo con fines maliciosos. Obviamente esto puede ser evitado sin entrar nunca al estado conectable,

3 Es cuando ambas partes en una comunicación comparten la misma clave secreta y su envío se realiza de tal forma que no permita revelar ningún dato acerca de la clave a entidades ajenas.

4 Proceso por el cual dos dispositivos establecen una clave compartida que pueden usar cuando se vuelvan a contactar.

pero de esta manera no se podría establecer ninguna conexión Bluetooth. El proceso de identificación de un dispositivo se provee a través del mecanismo de autenticación Bluetooth, que es un esquema denominado de desafío-respuesta, donde el dispositivo verificador envía un desafío aleatorio al dispositivo solicitante y queda a la espera de una respuesta válida. Este proceso es sólo de una sola mano y si se necesita que sea mutuo, debe repetirse con los roles cambiados.

La encriptación *broadcast* tiene un problema debido al paradigma punto a punto utilizado en Bluetooth; un dispositivo esclavo, además de sí mismo, sólo está pendiente del dispositivo maestro de la piconet, por lo que no se tiene ningún tipo de seguridad respecto del resto de los esclavos de la red. Cada enlace en la piconet usa diferentes claves de encriptación, y que están basadas en sus claves de enlace respectivas. Si el maestro desea enviar un mensaje encriptado a todos sus esclavos, puede hacerlo utilizando mensajes direccionados individualmente (mensajes unicast), lo que introduciría una sobrecarga innecesaria. Una mejor alternativa sería que el maestro modifique todas las claves de enlace a una clave temporaria: la clave maestra. Luego, en base a ella, todos los dispositivos son capaces de generar una clave de encriptación común que puede ser utilizada en transmisiones *broadcast* que direccionen a todos los esclavos de forma simultánea.

En conclusión, existen tres modos primarios de seguridad:

- MODO 1:

  Todos los mecanismos de seguridad (autenticación y encriptación) se encuentran deshabilitados. Además el dispositivo permite cualquier tipo de conexión Bluetooth a él.

- MODO 2:

  Los mecanismos de seguridad se inician luego de haberse establecido un canal entre los niveles de los protocolos LMP y L2CAP. El acceso a los servicios y dispositivos los controla un gestor de seguridad, su interface es muy sencilla y no requiere de ninguna codificación adicional de PIN o claves.

- MODO 3:

  Se inician todos los mecanismos de seguridad antes que se establezca un canal de conexión. Además de la encriptación tiene autenticación PIN y seguridad MAC, y rechaza comunicaciones con dispositivos sin emparejar.

El modo más común y utilizado es el Modo 2, el cual es generalmente usado para proteger los servicios ofrecidos por un dispositivo Bluetooth. Sólo se invoca cuando se realiza la petición de un servicio en particular, o más específicamente, cuando se realiza una solicitud de conexión para establecer un enlace a una capa de protocolo específica.

Cuando se intenta establecer una conexión punto a punto en las capas L2CAP o RFCOMM, el gestor de seguridad se concentra en ellas como mediador. No importa si la conexión es iniciada por una aplicación propia o fue solicitada por un dispositivo remoto; el gestor tiene conocimiento de lo que está sucediendo y puede decidir en el momento basándose en la configuración almacenada en la base de datos del servicio. Las acciones posibles a tomar son:

- No hacer nada y permitir que se establezca la conexión.
- Iniciar el mecanismo de autenticación.
- Iniciar el mecanismo de autorización[5].
- Comenzar la encriptación una vez que se establezca el enlace.

El gestor de configuración puede, además, de acuerdo a la especificación Bluetooth, iniciar medidas de seguridad si la aplicación del dispositivo inicia un tipo particular de conexión. Se puede notar que el proceso de autenticación es soportado en solicitudes de conexión entrantes y salientes, mientras que los procesos de autorización y encriptación son disparados solamente en solicitudes entrantes.

2.10 Frecuencias, Bandas, potencias y estudios sobre la salud

Como se ha mencionado anteriormente, la frecuencia utilizada por Bluetooth es a través de una radiofrecuencia sobre la banda ISM (Industrial, Scientific and Medical), especificada en la siguiente sección, definida por la ITU[6] desde los 2.4Ghz hasta 2.483 Ghz. [14]

Las potencias máximas permitidas, dependiendo de la clase de transmisor, se pueden observar en la siguiente tabla:

5 Proceso por el cual se le garantiza el permiso al dispositivo que solicita acceso a los servicios ofrecidos.

6 Del inglés International Telecommunication Union (Unión Internacional de Telecomunicaciones) es el organismo especializado de la ONU que se encarga de regular las telecomunicaciones a nivel internacional entre las distintas administradoras y empresas operadoras.

Tabla 3: Potencias máximas permitidas para Bluetooth

Nota:

1 mW (millivatio) = 10^{-3} W

El dBm se define como el nivel de potencia en decibelios en relación a un nivel de referencia de 1 mW.

2.10.1 Banda ISM

La **ISM** son bandas reservadas internacionalmente para uso no comercial de radiofrecuencia electromagnética en áreas industrial, científica y médica. En la actualidad, estas bandas han sido popularizadas por su uso en comunicaciones WLAN (*por ej.* Wi-Fi) o WPAN (*por ej.* Bluetooth).

Las bandas **ISM** fueron definidas por la ITU en el artículo 5 de las Regulaciones Radio (RR), concretamente puntos 5.138 y 5.150 [15].

El uso de estas bandas de frecuencia está abierto a todo el mundo sin necesidad de licencia, respetando las regulaciones que limitan los niveles de potencia transmitida. Este hecho fuerza a que este tipo de comunicaciones tengan cierta tolerancia frente a errores y que utilicen mecanismos de protección contra interferencias, como técnicas de ensanchado de espectro (Regulaciones de Radio Art. 15.13) [16]

Los dongles utilizados en este prototipo están certificados por la Comisión Europea (CE) [17] y la Comisión Federal de Comunicaciones (FCC) [18].

2.10.2 Estudios sobre la salud, recomendaciones de la OMS (Organización Mundial de la Salud)

"El marco normativo argentino sobre los niveles de la Máxima Exposición Poblacional (MEP) a las Radiaciones No Ionizantes está basado en las últimas recomendaciones de la Organización Mundial de la Salud (OMS).

La OMS es el organismo encargado de orientar y coordinar los estudios científicos, estadísticos y epidemiológicos sobre todo lo que concierne a la protección de la salud y el medio ambiente, generados por los principales centros de investigación e instituciones científicas en el mundo, a partir de lo cual se realizan recomendaciones.

Es por ello que con el objeto de asegurar que la exposición humana a los campos electromagnéticos no tenga efectos perjudiciales para la salud y que los equipos generadores de esos campos sean inocuos para la salud, se han adoptado diversas directrices y normas internacionales.

La OMS basa sus recomendaciones en los estudios de la Comisión Internacional para la Protección contra las Radiaciones No Ionizantes (ICNIRP). En particular con respecto a la telefonía celular, la información científica producida hasta el momento por la ICNIRP, no indica la necesidad de algún tipo de precauciones que se deban sumar a las recomendaciones de la OMS para el uso de teléfonos móviles, o la instalación de las antenas que permiten dar cobertura a este servicio radioeléctrico.

Hasta la actualidad, dentro de los límites recomendados por la OMS, no existen evidencias científicas que permitan afirmar fehacientemente que las RNI (Radiaciones No Ionizantes) produzcan efectos adversos sobre la salud de la población. Por el momento, el único efecto fehacientemente comprobado, cuando se sobrepasan dichos límites recomendados por la OMS, es el calentamiento de los tejidos, el cual desaparece un tiempo después de quitar la fuente de radiación, tal como ocurre con cualquier fuente de calor convencional." [19]

2.10.3 Recomendaciones de la OMS

"Existen normas establecidas para proteger nuestra salud, como las relativas a

aditivos alimentarios, a las concentraciones de productos químicos en el agua o a los contaminantes del aire. De forma similar, existen normas que previenen la exposición excesiva a los campos electromagnéticos presentes en el entorno" *[20]*

En la siguiente tabla se detallan las frecuencias de exposición más comunes junto con sus límites máximos recomendados con su correspondiente unidad:

	Frecuencia de la red eléctrica europea	**Frecuencia de estaciones base de telefonía móvil**		**Frecuencia de los hornos de microondas**	
Frecuencia	**50 Hz**	**50 Hz**	**900 MHz**	**1,8 GHz**	**2,45 GHz**
	Campo eléctrico (V/m)	Campo magnético (μT)	Densidad de potencia (W/m2)	Densidad de potencia (W/m2)	Densidad de potencia (W/m2)
Límites de exposición para la población	5 000	100	4,5	9	10
Límites de exposición ocupacionales	10 000	500	22,5	45	

Tabla 4: Resumen de los límites de exposición recomendados por la ICNIRP [20]

En la última columna donde se especifican los valores para las frecuencias de microondas es que se observa utilizada por Bluetooth (2.4Ghz.) con su valor límite de exposición (densidad de potencia) de 10 W/m2 (Watts sobre metro cuadrado).

3 Software Libre

La palabra “libre” tiene diferentes significados, y sólo uno de ellos se refiere a precio. Una de las definiciones más importantes deriva del término “libertad de expresión” o aún mejor de la expresión “trabajo libre no forzado”. No se refiere a libre como gratuito, sino a libre en el sentido de limitado en cuanto a su control por los otros, lo que implica un control transparente y susceptible de modificación, de forma similar a cuando las leyes de una sociedad libre son de tal forma que hacen su control comprensible y abierto a la modificación. El objetivo del movimiento del software libre es producir código en la medida en que pueda ser transparente y susceptible de modificación.

Entonces se puede decir que el **software libre** es una cuestión de libertad y no de precio; a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Por lo tanto se podría tener como referencia a cuatro clases de libertad para los usuarios de software:

1. **Libertad 0:** Libertad para ejercitar el programa sea cual fuere nuestro propósito.
2. **Libertad 1:** Libertad para estudiar el funcionamiento del programa y adaptarlo a las necesidades del usuario. *
3. **Libertad 2:** Libertad para redistribuir copias y ayudar de esta manera al resto de los usuarios.
4. **Libertad 3:** Libertad para mejorar el programa y luego publicarlo para el bien de los demás. *

* *para estas clases es indispensable tener acceso al código fuente*.

Por consiguiente, software libre es cualquier programa cuyos usuarios gocen de estas libertades, de modo que deberían ser libres de redistribuir copias, con o sin modificaciones, de forma gratuita o remunerada a cualquiera y en cualquier lugar. [21]

La libertad para usar un programa significa que cualquier persona u organización podrá ejecutarlo desde cualquier sistema informático con cualquier fin y sin la obligación de comunicárselo ni al desarrollador ni a ninguna entidad en particular, por lo cual no debe ni pedir permiso ni pagar para ello. La libertad de redistribución de copias supone incluir las formas binarias o ejecutables del programa y el código fuente tanto de las versiones

originales como de las modificadas. No sucede nada si no se puede producir una forma ejecutable, ya que no todos los lenguajes de programación la soportan, pero se debe tener la libertad de redistribuir tales formas si existe el modo de hacerlo.

El software libre no significa que sea “no comercial”, ya que cualquier programa libre se encontrará libre para su uso, desarrollo y distribución comercial.

3.1 Copyleft

El copyleft es una forma de licencia que permite que una obra o trabajo, tal como un programa de software, un documento, un tema musical o una obra de arte, pueda ser utilizada para modificar el derecho de autor de la misma, ya que dicha metodología de desarrollo exige que todas las versiones modificadas y extendidas de la misma sean también libres. [22]

El término *copyleft* nació como una derivación de *copyright*, jugando con los significados en inglés de *right*: derecho como acepción jurídica y derecha como acepción política. De esta manera, el *copyleft* sería la reivindicación de la libertad frente a los derechos de autor que la coartan. A la vez también *left* se asocia con el participio del verbo *to leave* (dejar) atribuyendo a la difusión con esta filosofía de “dejar” a disposición de otros usuarios para que posteriormente puedan usarla libremente de manera indefinida.

Para cubrir un programa con *copyleft*, primero se debe declarar que sus derechos están reservados, es decir que tiene *copyright*. Luego, deben agregarse unos términos de distribución que serán un instrumento legal para utilizar, modificar y redistribuir el código del programa o de cualquier programa derivado del mismo, pero solamente si los términos de distribución no son alterados. De esta manera, el código y las libertades se hacen legalmente inseparables. El *copyleft* es entonces una forma de usar los derechos de autor de un programa, y que no implica abandonar dichos derechos, ya que sino el uso del copyleft sería imposible.

Si se pone un programa bajo dominio público, sin derechos de autor, se estaría permitiendo que la gente lo pueda compartir junto con sus mejoras, si lo desean; pero también se estaría permitiendo que gente no tan cooperativa convierta al programa en software privativo, logrando de esta manera que las personas que reciban dicho programa con esas modificaciones no tengan la libertad que le dio el autor original, ya que ha sido eliminada por el intermediario. Justamente por este motivo es que se protege al programa

con *copyleft*, ya que de esta manera, cualquiera que redistribuya el software modificado o sin modificar, deberá otorgar al usuario la libertad de copiarlo y modificarlo para garantizar así que se mantendrán estas libertades para todos los usuarios.

3.2 Categorías de software

El siguiente diagrama muestra de alguna manera las distintas categorías que existen de software:

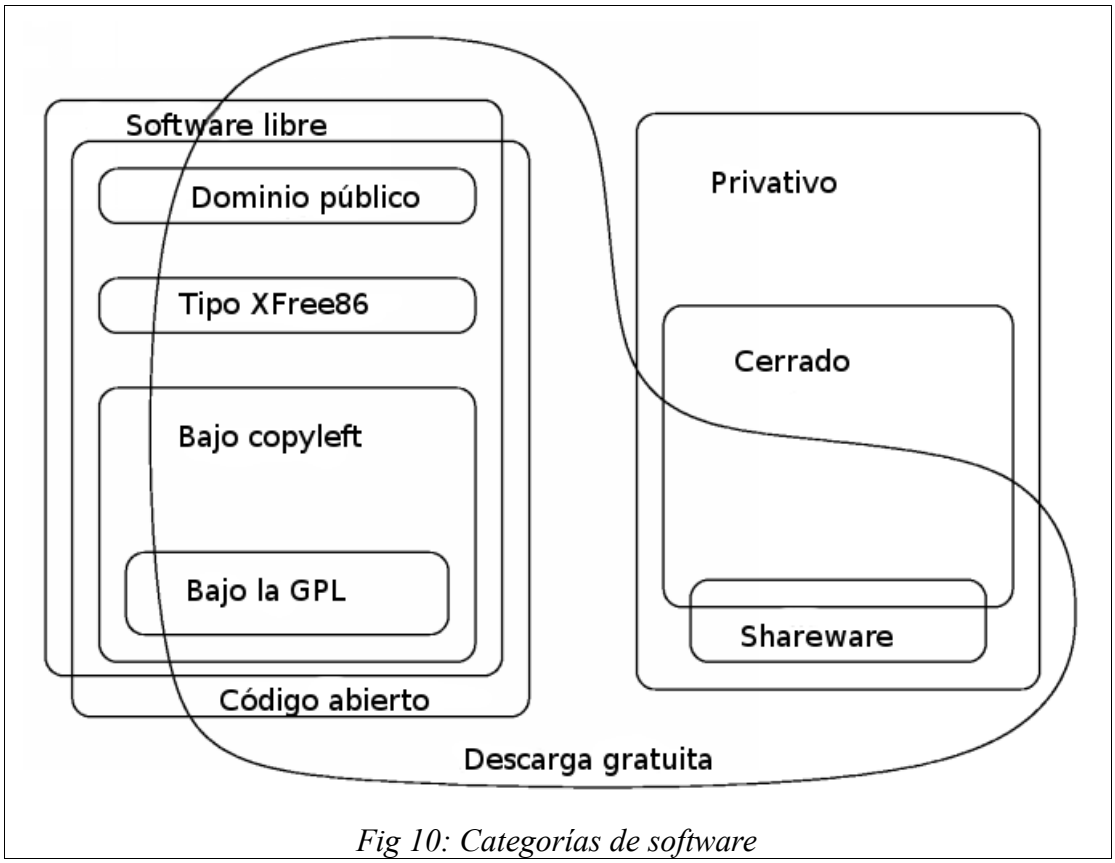


Fig 10: Categorías de software

3.2.1 Software Libre:

Se refiere al software con autorización para que cualquiera pueda usarlo, copiarlo y distribuirlo, ya sea con o sin modificaciones, gratuitamente o remunerado. O sea que el código fuente debe estar disponible. Este tipo de software es mayormente más fíable que el no libre.

3.2.2 Software de Código Abierto (Open Source):

Es similar al software libre pero con la diferencia de que el mismo se enfoca más a cuestiones prácticas cuando proporciona el código fuente a los usuarios mientras que el software libre aborda el problema desde el punto de vista ético referido a las libertades de los usuarios. Además tienen una leve diferencia respecto a las licencias que proveen, siendo el open source un poco más restrictivo que el software libre. Sin embargo, sus diferencias son pocas y se puede decir que casi todo el sofware libre es de código abierto y casi todo el software de código abierto es libre.

3.2.3 Software de Dominio Público

Se trata de software que no está protegido por derechos de autor. Es un caso especial de software libre no protegido con copyleft, lo que quiere decir que algunas copias o versiones modificadas pueden no ser completamente libres.

3.2.4 Software protegido con copyleft

Es software libre cuyos términos de distribución aseguran que todas las copias de todas las versiones son software libre, lo que prohíbe, por ejemplo, que otros agreguen requisitos adicionales, y exige que el código fuente sea público. Existen varias formas posibles de escribir términos de distribución copyleft (licencias), aunque en la práctica casi todo software copyleft usa la Licencia Pública General (GPL) de GNU, la cual fue creada en 1989 y está orientada a proteger la libre distribución, modificación y uso de software.

3.2.5 Software libre no protegido con copyleft (tipo XFree86)

Este tipo de software implica que algunas copias o versiones modificadas del mismo puedan no ser completamente libres. Por ejemplo, una compañía de software podría compilar el programa, con o sin modificaciones, y distribuir el archivo ejecutable como si fuera software privativo.

3.2.6 Software cubierto por la GPL

Es software que tiene licencia GPL (General Public License), que como se dijo anteriormente, es un conjunto específico de términos de distribución que se emplea para proteger un programa con copyleft. Un ejemplo que use esta licencia es el Proyecto GNU[7], que es un sistema operativo similar a Unix pero completamente libre y que incluye todo el software GNU, además de muchos otro paquetes que pueden no ser GNU.

3.2.7 Software Privativo

Es software que no es libre ni semilibre[8], lo que hace que su uso, redistribución o modificación estén prohibidos, ya que requieren de una autorización o bien está tan restringido que no pueden hacerlo libremente.

3.2.8 Software Cerrado

Se trata de software a medida desarrollado para un usuario, que por lo general es una organización o empresa, el cual lo posee, lo utiliza y no lo libera al público ni como código fuente ni como binario. Se dice que es cerrado porque quien lo adquiere no tiene la posibilidad de corregirlo o modificarlo, ni tampoco puede estudiar su comportamiento. El código fuente permanece oculto y sólo puede accederlo la empresa que desarrolla y comercializa el producto. Prácticamente todo el software convencional comercial es cerrado, como por ejemplo los sistemas operativos Windows.

3.2.9 Shareware

Por último el shareware es software que se permite redistribuir copias de él pero que por cada copia utilizada, el usuario debe pagar un cargo por licencia. Por lo tanto, no es software libre ni semilibre, ya que el código fuente no está disponible y además no se puede hacer una copia e instalarla sin pagar.

---

7 Acrónimo recursivo que significa GNU's Not Unix, es decir, GNU No es Unix.

8 Aquellos que incluyen autorización para usarlo, copiarlo, distribuirlo y modificarlo pero sin fines de lucro y con la restricción de no poder utilizarlo en un sistema operativo libre.

3.3 Plataforma Linux

Una **plataforma** es la base sobre la cual un programa o aplicación puede ejecutarse y/o desarrollarse, es decir, que define un estandar por el cual un sistema puede ser desarrollado. Si bien no se refiere a un sistema operativo, dicho término se suele utilizar para ello. Ejemplos de plataformas son x86 (basadas en Intel, AMD, etc.), Sparc, PowerPC, Alpha, ARM, etc.

El sistema **GNU/Linux** es una combinación del núcleo (o kernel) de un sistema operativo similar a Unix, llamado Linux, y herramientas de sistema GNU, tales como ambientes de trabajo (shells) compiladores, editores, formateadores de texto, correo, etc. Generalmente a la unión de ambos proyectos se los suele denominar simplemente Linux, pero en realidad son dos cosas que se iniciaron de forma separada pero que luego se unieron para formar un sistema operativo. Se dice que es Open Source porque su código fuente puede ser utilizado, modificado y redistribuido libremente bajo los términos de la licencia GPL y otras licencias libres.

Respecto a las plataformas, Linux es un sistema con plataformas múltiples, ya que puede ejecutarse tanto sobre x86 como PowerPC, Sparc y otras tantas plataformas. Otras características de Linux es que es multitarea, multiusuario y multiprocesador, que como sus nombres lo indican, se pueden ejecutar diferentes aplicaciones a la vez sin que interfieran entre ellas, diversos usuarios pueden trabajar en el mismo equipo al mismo tiempo y se puede ejecutar en más de un procesador. A Linux se lo considera un sistema fiable, sólido y potente. Incluso puede ejecutarse con muy pocos recursos en equipos de muy bajo rendimiento.

Por otro lado, Linux ha sido desarrollado por un grupo multidisciplinario de personas ubicadas en distintas partes del mundo, y no como el sistema operativo Windows, que fue desarrollado por una empresa como Microsoft, cuya licencia limita su uso al usuario final que lo compre. En cambio Linux, al ser software libre, no posee estas restricciones y permite instalarlo en cuantos equipos sean necesarios y su licencia permite modificar su código fuente según las necesidades de cada usuario y luego redistribuirlo. [23]

Otro punto fuerte de Linux es su seguridad y robustez. Es muy **seguro** debido a que la gran mayoría de los ataques de hackers son dirigidos a sistemas operativos Windows, y además, al disponer del código fuente, cualquiera puede darse cuenta de algún fallo y comunicarlo al resto de los usuarios. Y es **robusto** porque puede pasar

mucho tiempo sin la necesidad de apagar o reiniciar el equipo, y además, si una aplicación falla, no se bloquea totalmente el equipo, ya que al tener separados los modos de ejecución de los procesos que conforman los programas, un proceso de usuario no interfiere con los procesos del kernel[9].

3.3.1 Distribuciones

Una **distribución** es una recopilación de software ya compilado y empaquetado para facilitar su instalación y configuración, y permite instalar un sistema GNU/Linux completo. La mayoría de las distribuciones propone también su propia instalación gráfica así como un sistema de administración de paquetes que permite la instalación automática de software por medio de la administración de dependencias, que son vínculos a librerías externas.

Existen varias distribuciones disponibles: Debian, Suse, RedHat, Knoppix, Mandriva, Slackware, etc. Cada distribución tiene sus ventajas y sus desventajas, dependiendo del tipo de usuario al que esté orientado.

La distribución elegida para esta tesis es la **Debian** por diversos motivos:

- Es 100% libre.
- Es de muy buena calidad, ya que antes de liberar una nueva versión, muchos usuarios prueban el software durante un largo período para poder reportar posibles defectos.
- Tiene una gran cantidad de software (muchos paquetes realizados por miles de desarrolladores en el mundo).
- Posee un excelente sistema de empaquetamiento del software, con un buen control de dependencias y conflictos, y una fácil actualización por diferentes medios.

3.4 Python

A la hora de elegir un lenguaje de programación, **Python** es una muy buena opción; primero, porque es fácil, de código corto y legible, y segundo, porque se trata de un lenguaje orientado a objetos muy poderoso y dinámico, y en el que permite focalizarse más en el algoritmo en sí sin tener que preocuparse por problemas de manejo de memoria u otros aspectos ajenos al objetivo del programa. [24]

9 Núcleo de un sistema operativo. Es el encargado de que el software y el hardware puedan trabajar conjuntamente.

3.4.1 Características del lenguaje

Python es un lenguaje similar a Perl[10], pero con una sintaxis muy limpia y que favorece a la legibilidad del código. Se trata de un lenguaje interpretado, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos.

Que sea **interpretado** quiere decir que se ejecuta utilizando un programa intermedio en vez de compilar el código a lenguaje máquina para ser ejecutado directamente en la computadora. La ventaja de que sea interpretado es su flexibilidad y portabilidad, aunque puede ser más lento que los compilados. De todas maneras, en la práctica, Python es un lenguaje semi-interpretado, ya que el código fuente se traduce la primera vez que se ejecuta a un pseudo-código máquina intermedio (bytecode) que es el el que se ejecutará cada vez que se corra el programa.

Que sea de **tipado dinámico** significa que no es necesario declarar el tipo de cada variable que se utiliza ya que el mismo se determinará en el momento de la ejecución del programa y además puede ir cambiando su tipo.

Al ser **fuertemente tipado** no se permite tratar a una variable como si fuera de otro tipo al que tiene asignado, por lo que se requiere convertir de manera explícita dicha variable al nuevo tipo.

Es **multiplataforma** porque existe un intérprete para muchos tipos de plataformas, como Linux, Windows, Mac OS, OS/2, Solaris, lo que permite que pueda ser ejecutado en todos ellos sin la necesidad de alterar el código fuente del programa.

Y por último se dice que es **orientado a objetos** ya que permite abstraer el problema del mundo real a la modelización con clases y objetos.

Otras características adicionales de Python es que permite programación imperativa, funcional y orientada a aspectos, por lo que se dice que abarca la mayoría de los paradigmas de programación existentes.

3.4.2 Extensión para Bluetooth

Como Python no trae soporte para Bluetooth en su distribución estándar, se agrega una extensión para tener acceso al recursos de esta índole. Uno de los más usados es PyBluez, que es compatible con Windows y GNU/Linux, pero en el primero es menos funcional ya que no provee una API[11] para las capas L2CAP y HCI. Además requiere de la

10 Lenguaje de programación de alto nivel, de propósito general, interpretado y dinámico creado en 1987 y basado en varios lenguajes como C y shell scripting (sc). Su mayor ventaja es el poder de procesamiento de archivos de texto.

11 Del inglés *Application Program Inteface*, es el conjunto de procedimientos, funciones y/o métodos que ofrece una

pila de Microsoft Bluetooth y no funciona con otras pilas Bluetooth que no sean provistas por el ambiente Widcomm[12]. Otra gran desventaja es que la pila de Windows no soporta tener más de un sensor Bluetooth conectado, por lo que para nuestro caso es muy necesario y debido a esto hemos optado por trabajar bajo GNU/Linux.

3.4.3 Persistencia

Para poder persistir los datos es conveniente elegir un ORM (del inglés Object Relational Mapper) que permita traducir tablas de bases de datos relacionales en objetos cuya manipulación es transparente en la base.

El ORM elegido para uno de los módulos de esta tesis es el SQLObject, el cual permite especificar en la declaración de las clases cómo la misma se traduce a una tabla en la base de datos. Este ORM produce código para acceder a la base y actualizarla fácilmente. También posee una característica que evita tener que escribir consultas SQL textuales.

3.5 Django

Django es un framework de desarrollo web que ahorra tiempo y facilita el mantenimiento de aplicaciones web con un mínimo esfuerzo. Provee abstracciones de alto nivel de los patrones de desarrollo web más comunes, atajos a tareas de programación frecuentes y convenciones claras de cómo resolver problemas. [25]

Un framework web provee una infraestructura de programación para las aplicaciones que le permite al programador focalizarse en escribir código limpio, mantenible y sin tener que “reinventar la rueda”, es decir, sin la necesidad de programar soluciones que ya son conocidas y necesarias para muchos casos en general.

Django nació en el año 2005 y su nombre se debe al guitarrista de jazz Django Reinhardt. En la actualidad es un proyecto *open source* con colaboradores en todo el mundo. Al haber nacido en un ambiente de noticias, Django ofrece varias características que se adaptan muy bien a sitios de contenido que ofrecen información dinámica gestionada en una base de datos. Es por eso que uno de los puntos más fuertes de este *framework* es su interface de administración, que es una parte esencial en la infraestructura en donde los administradores del sitio pueden agregar, modificar y eliminar contenido. Las interfaces de administración son siempre iguales: autenticación de

librería para ser usada por otro software de una forma más abstracta.

12 Primer pila Bluetooth para sistemas operativos Windows y que luego fue adquirida por Broadcom Corporation.

usuarios, visualización y manejo de formularios, validación de los datos ingresados, etc. Son todas tareas repetitivas que Django las resuelve en pocas líneas de código y de una forma muy eficiente.

3.5.1 Características Principales

La principales características de Django son las siguientes:

- Es un framework escrito 100% en Python.
- Posee un modelo de datos simple a través de clases y que con su ORM es posible generar la tablas de la base de datos de manera directa y sin necesidad de manipular sus datos utilizando el lenguaje SQL.
- Genera automáticamente una interfaz de administración para el modelo creado a través del ORM.
- Genera automáticamente los formularios necesarios a través del modelo de datos.
- Utiliza URLs elegantes y de manera simple basadas en expresiones regulares.
- Contiene un sistema de plantillas poderoso y con herencia en el que se pueden separar el diseño del contenido y del modelo (patrón MVC [13]).
- Tiene un sistema de caché integrado.
- Soporta internacionalización incluyendo traducciones incorporadas de la interfaz de administración.
- Sigue lo que se denomina el principio DRY (Don't Repeat Yourself) cuyo objetivo es definir el modelo de datos en un solo lugar y derivar cosas automáticamente a partir de él, reduciendo la duplicación de código.

Una característica muy particular de este framework es que no es necesario utilizar todo lo que trae, es decir, no es obligatorio usar el sistema de plantillas que trae ni la API de base de datos, pudiendo utilizar otro sistema de plantillas así como también utilizar otra capa de abstracción de datos (lectura de archivos XML o de otro tipo). Esto se debe a que cada pieza de Django (modelos, vistas, plantillas) está desacoplada del resto.

13 El MVC es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz del usuario y la lógica de control en tres componentes diferentes.

4 Proyecto Blue4AS (Bluetooth For Accessibility Systems)

4.1 Motivación

Las aplicaciones ubicuas y sensibles al contexto abarcan un amplio rango que comprende sistemas de monitoreo de transporte, aplicaciones turísticas, entretenimiento y todo el enorme espectro de aplicaciones de "knowledge management"[14], en particular comunidades de conocimiento en ciertas áreas específicas (tales como investigación, finanzas, etc).

Al igual que en otras áreas, los avances en tecnologías de hardware y comunicaciones no han sido debidamente acompañados por un avance semejante en modelos, métodos, formalismos y procesos para desarrollar software que responda tanto a las necesidades de los usuarios como que sea capaz de explotar las posibilidades de la tecnología.

Uno de los aspectos desarrollados en este trabajo ha sido generar herramientas de software que permitan realizar una comunicación lo suficientemente genérica para admitir diferentes configuraciones sin la necesidad de instalar una aplicación cliente en el dispositivo móvil. El software y hardware utilizado en marketing por proximidad es bastante acotado y cerrado. Es por esto que resulta importante realizar una implementación genérica para cubrir un amplio espectro de espacios de alta concurrencia y necesidades.

Para que los servicios brindados se encuentren cada día más cerca de las personas con capacidades diferentes, la tecnología debe proveer una amplia gama de servicios generando una importante experiencia para aquellos que no tienen la posibilidad de ver o escuchar.

Se decidió llamar al proyecto Blue4AS, por sus siglas en inglés "Bluetooth For Accessibility Systems" (Bluetooth para Sistemas de Accesibilidad), ya que se hizo hincapié en que sea accesible para todos y además se logra un doble sentido fonético –

14 Del inglés la "gestión del conocimiento" es un concepto aplicado en organizaciones para transferir el conocimiento y la experiencia que hay entre sus miembros para poder ser utilizado por otros miembros de la organización.

del inglés *for us*, “para nosotros” – .

4.2 Desarrollo propuesto

El prototipo cuenta con dos módulos principales: por un lado el sistema de administración remota el cual, a través de una interfaz gráfica (Web), permite configurar qué contenidos se desean transmitir mediante los sensores Bluetooth diseminados por todo el edificio o recinto; y por otro lado se tiene al sistema ubicuo, que es el encargado de la comunicación con los dispositivos móviles a través de su red de sensado y de la recolección de información. Dichos datos son almacenados para futuras estadísticas y monitoreo en tiempo real. Cuando los receptores se coloquen en el radio de alcance de los sensores estratégicamente ubicados, se podrá determinar en qué sector se encuentra y en base a eso, transmitir la información precisa correspondiente a dicha ubicación. Se almacena la información de qué dispositivo se encuentra en cada sector en un momento determinado. Estos datos permiten realizar estadísticas, como ser sectores más concurridos, cantidades estimadas de dispositivos por franjas horarias, cantidad de dispositivos en tiempo real de todos los sectores sensados, etc. Además, al contar con esta información, también evitamos retransmisiones indeseables a un mismo dispositivo.

La arquitectura elegida para el módulo de carga y configuración del sistema es sobre una plataforma Web de manera que sea fácilmente accesible desde cualquier punto. Y para el sistema ubicuo de emisión y recolección de información se optó por un servicio o proceso *background*[15] ejecutado sobre una estación y una red de sensores Bluetooth conectados a él. Los protocolos utilizados en las comunicaciones vía Bluetooth son: SDP (cap. 2.2.7) para el descubrimiento de dispositivos, RFCOMM en la utilización de sockets (cap. 2.2.6) y Object Push (cap 2.3.3) para el envío de la información a los dispositivos.

4.3 Requerimientos

Una de las características buscadas de este prototipo es que tenga bajos requerimientos para poder implantar el mismo. Una posible implementación mínima del sistema es a través de una PC de bajo poder de cómputo, con puertos USB capaces de albergar a los *Dongles* o sensores Bluetooth que serán los encargados de la transmisión

15 Así se denomina cuando un proceso se realiza en segundo plano.

hacia los dispositivos móviles y otra (que podría ser la misma PC) con la capacidad de alojar un servidor Web y un motor de Bases de Datos. La potencia de cómputo necesaria para que el funcionamiento del sistema sea exitoso es realmente baja, ya que cualquier PC de hoy en día (e incluso PCs de hace algunos años) es altamente capaz de soportar este tipo de procesamiento.

Para una mejor performance del funcionamiento general del sistema, este proyecto está especialmente desarrollado de manera tal que se puedan hacer envíos simultáneos por diferentes sensores a distintos dispositivos, o bien, tener un sensor dedicado al continuo descubrimiento y otros N sensores dedicados a las transmisión simultánea. Para obtener una performance de este tipo va a ser necesario, además de una mayor cantidad de sensores, que el sistema sea montado sobre GNU/Linux o similar, ya que la implementación de la pila Bluetooth permite más de un dispositivo conectado a una PC.

Esta fue una característica deseada del proyecto debido a que la idea es realizar implantaciones del sistema sin la necesidad de contar con equipamiento costoso. De esta manera, el factor económico no sería un impedimento de la puesta en marcha del proyecto.

4.4 Arquitectura utilizada

En el sistema ubicuo se utiliza un esquema en donde una PC de baja potencia posee sensores – dongles - Bluetooth capaces de realizar la transmisión a dispositivos móviles circundantes. Como se puede observar en la Fig. 11, un cliente o estación puede tener tantos sensores como se desee (se pueden agregar Hubs USB o placas PCI para incrementar la cantidad de puertos locales). Cuantos más sensores se tenga por zona, más veloz será la transmisión a múltiples dispositivos móviles. El sistema es fácilmente escalable, simplemente agregando más dongles a una estación, adaptando su funcionamiento de manera automática en base a la cantidad de dongles provistos.

Fig 11: Cliente (dongle station) con 3 sensores Bluetooth conectados a un Hub USB de 7 puertos

La estación huésped de los sensores, a su vez, se comunica con el motor de bases de datos para obtener todos los parámetros de configuración adecuados y la información que le corresponde para enviar por cada zona que tenga registrada. Dicha estación es la encargada de enviar a todo lo que encuentre a su alcance y de registrarlo en la base de datos.

El proyecto fue desarrollado para que no haya necesidad de contar con una PC por zona en la que se quiera transmitir, sino que se podría tener una sola estación capaz de albergar sensores que estén diseminados en diferentes lugares o zonas, en donde en cada una se podría enviar información diferente discriminando justamente por la zona en la que se encuentre.

El módulo de administración remota actúa bajo una arquitectura típica cliente/servidor donde este último se conecta a una base de datos para interactuar y realizar las configuraciones pertinentes.

4.5 Aplicación

4.5.1 Escenario típico

En primera instancia se debe configurar el sistema mediante la Administración remota (cap. 4.5.3) indicando mínimamente las zonas y los sensores asociados a cada una de ellas y la información que se desea transmitir por cada área. Una vez finalizado este primer paso y con el sistema en marcha, los usuarios que pasan por el alcance de cada sensor son detectados y se le ofrecen los datos que se asociaron en esa zona; el usuario puede aceptar o rechazar dicho servicio. El envío de la información va a depender de ciertos mecanismos empleados por el sistema descriptos en esta sección. Dicho comportamiento es almacenado para su posterior estudio. Los usuarios irán transitando por las diferentes zonas de detección y se le enviarán los datos correspondientes. De esta manera se puede estudiar el flujo y tránsito de las zonas, creando dos objetivos: el envío de información y la adquisición de conocimiento.

4.5.2 Sistema Ubicuo

En esta sección se explica el funcionamiento general del sistema, diferentes mecanismos y políticas utilizadas, así como también se mencionan y detallan las clases más relevantes del modelo del módulo ubicuo y cómo interactúan entre ellas. De las clases planteadas sólo se muestra el conocimiento y comportamiento más relevante para la comprensión del mismo.

4.5.2.1 Funcionamiento

Como se ha mencionado anteriormente, el proyecto consta de dos interfaces o módulos que interactúan entre sí mediante una base de datos. Los aspectos más relevantes del sistema ubicuo se detallan a continuación.

Implementa un servicio o proceso background que se ejecuta constantemente en la estación utilizada para el envío de la información a través de sus sensores. El mecanismo es de la siguiente manera: al iniciar el sistema, se toman los valores necesarios para iniciar el proceso, como ser los datos para la conexión a la base de datos, de su archivo de configuración (cada estación tiene el propio) y luego establece la conexión con la base

de datos. De allí obtiene toda la información restante necesaria para el inicio y recupera los sensores que tiene asociados, el rol de cada uno de ellos (descubridor y/o transmisor) de todas las ubicaciones que tiene asociadas y la información a enviar por cada zona.

Una vez iniciado el proceso mediante sus descubridores (explicado en la Fig. 12), se comienza con la búsqueda concurrente[16] de dispositivos remotos (en el caso de que tenga asociado más de un sensor con el rol descubridor).

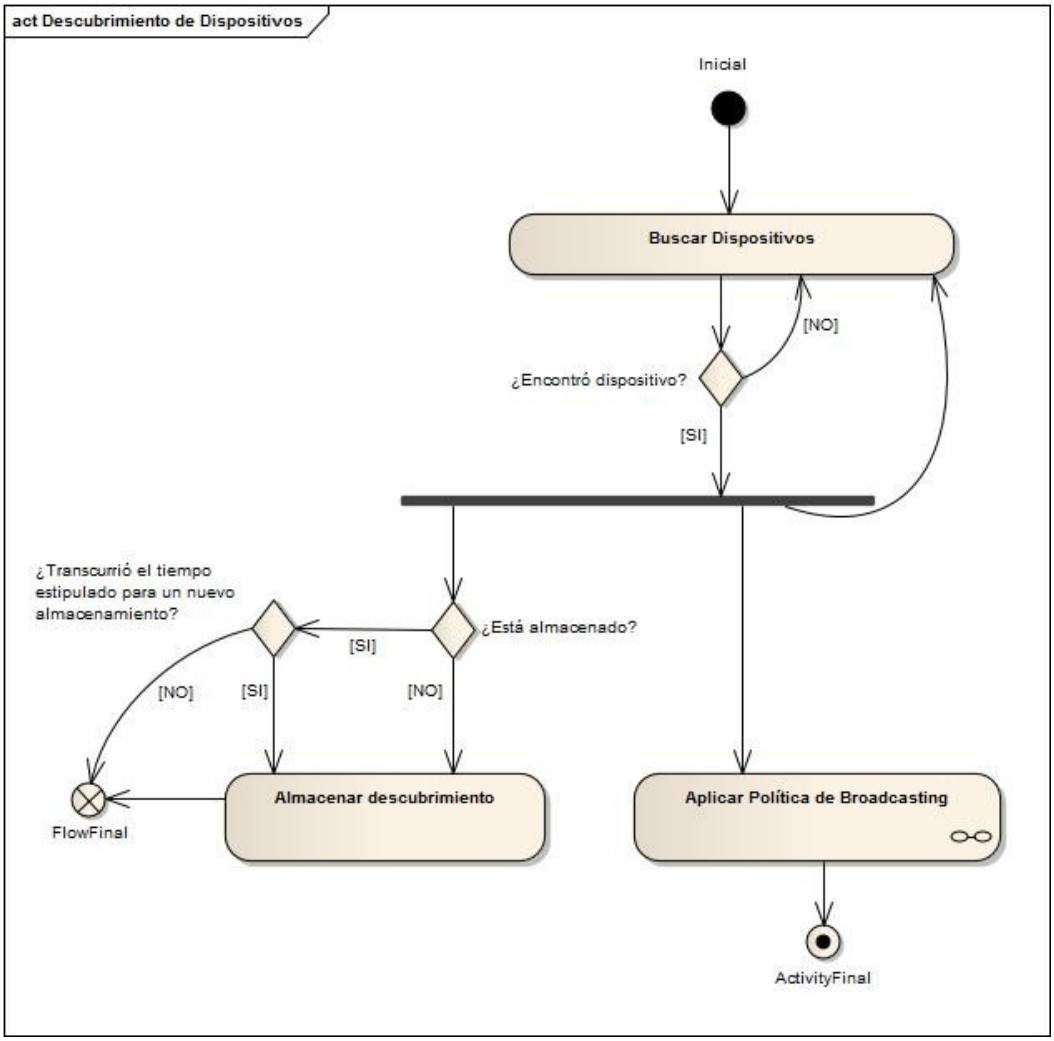


Fig 12: Proceso de descubrimiento de dispositivos

El sistema queda en espera hasta que alguno de los descubridores avise al objeto **SensorManager** (esta clase es la encargada de administrar el módulo ubicuo) que encontró algún dispositivo (indicando también la ubicación en donde se lo encontró) y

16 Se refiere a una búsqueda eficiente mediante procesos que se ejecutan simultáneamente.

sigue con su búsqueda.

Una vez encontrado un dispositivo remoto, se realizan dos tareas sobre el mismo: la primera consta en almacenar el registro del hallazgo. Si un dispositivo permanece mucho tiempo al alcance de un sensor, el mismo lo registrará unas 8 o 9 veces por minuto como mínimo; por este motivo, se implementó un mecanismo por el cual desde la administración se puede indicar un umbral mínimo (en minutos) de registro de los dispositivos y así evitar sobrecargar la base de datos. La segunda tarea a realizar es verificar concurrentemente la situación de cada dispositivo en particular en cuanto a la información a enviar. Es decir, que como hay descubridores constantemente buscando dispositivos, es indeseable que se envíe repetidamente la misma información una y otra vez al mismo dispositivo causando un malestar en el cliente y la cancelación o rechazo del servicio brindado. Es por ello que se implementaron diversos mecanismos para evitar la saturación de envío de información a los dispositivos móviles - políticas de broadcasting, cantidad de intentos ante fallas (*tries_on_failure*) y tiempo transcurrido desde la ultima detección y almacenamiento (*time_discovers_storage*) - .

Adicionalmente, el sistema provee un mecanismo de suscripción y bloqueo del dispositivo para la recepción o no de los datos por parte de los sensores. Dicho mecanismo es conocido como “política de broadcasting”, y se han definido dos posibles: de lista negra y de lista blanca (detalladas en el Cap. 4.5.2.2). Como se muestra en la Fig. 13, la recepción de los datos dependerá de la elección del usuario. El mismo puede suscribirse o desuscribirse al sistema, según la política configurada, realizando un cambio de estado mediante su dispositivo (simplemente cambiando el nombre a mostrar). Una vez modificado el mismo, se le enviará un mensaje de confirmación del cambio, y luego, si la decisión es la de recibir datos, continuará con el proceso de envío; y sino, el sistema no interactúa más con el usuario hasta un nuevo cambio de estado.

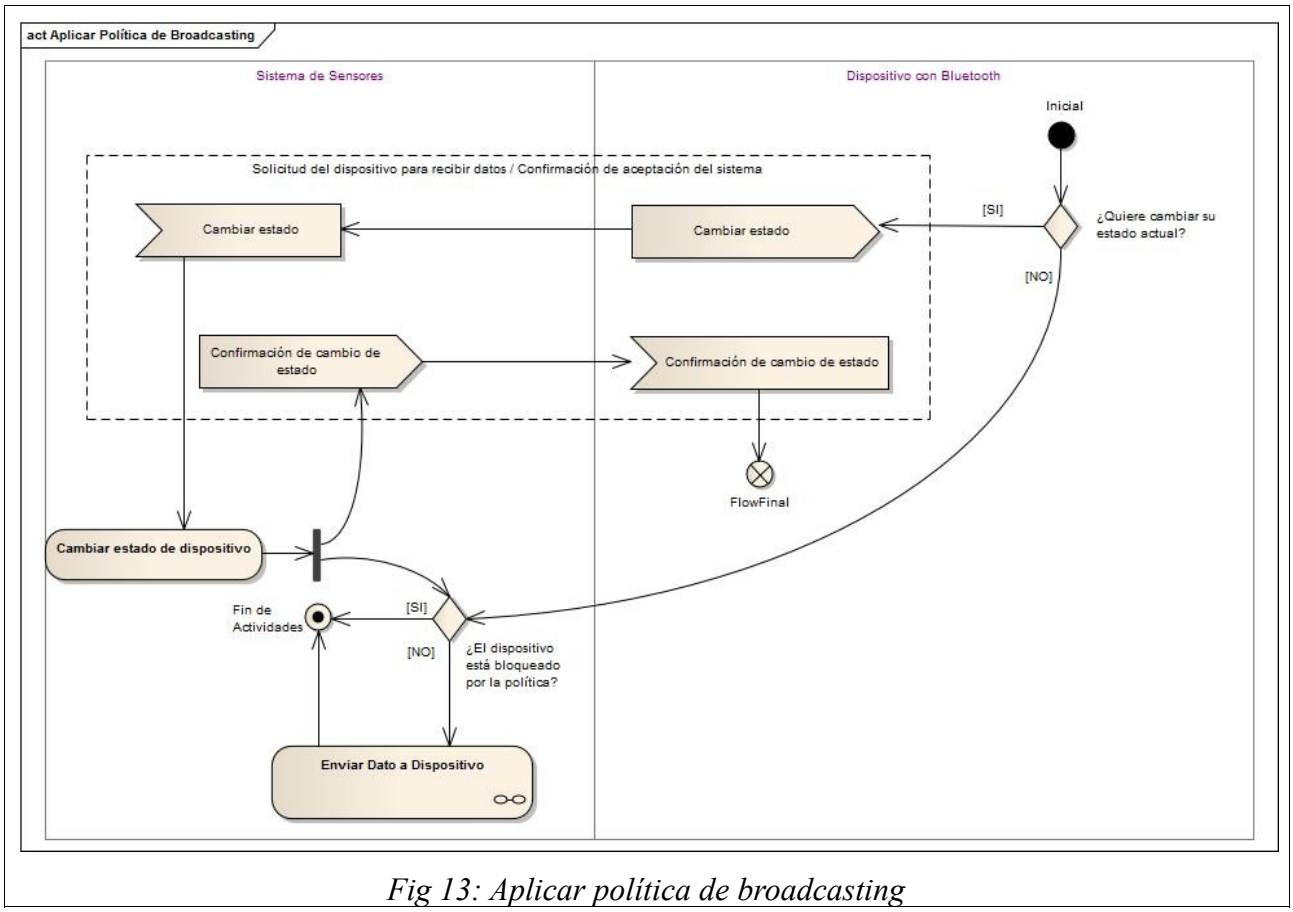


Fig 13: Aplicar política de broadcasting

Al agregar mediante la Administración Remota algún dato a enviar, además del nombre, el tipo y la localización del recurso, se configuran varios parámetros más, como ser:

- **Información habilitada (published):** Indica si el dato está o no habilitado para el envío.

- **Inicio de la publicación (start_publish_date):** Fecha y hora a partir de la cual se va a empezar a transmitir dicha información.

- **Fin de la publicación (end_publish_date):** Fecha y hora a partir de la cual se va a dejar de transmitir dicha información .

- **Cantidad de intentos ante falla (tries_on_failure):** Cantidad de intentos que va a realizar el sistema hacia un dispositivo en particular en el cual se haya producido un error o la cancelación de la transmisión.

- **Ubicación/es desde donde transmitir:** Se asocia el dato a una o a varias ubicaciones. Si un Descubridor encuentra un dispositivo en alguna de esas zonas, intentará enviar la información mediante los sensores configurados como transmisores en dicha zona.

- **Cantidad de días que pasaron antes de re-transmitir (days_to_repeat):** Esta información es útil en el caso en el que se desee periódicamente, con un intervalo de tiempo, transmitir algún dato que previamente ya había sido enviado al dispositivo encontrado.

Por lo tanto, dependerá de la configuración cargada mediante la Administración Remota si se va a enviar o no dicha información (Fig. 14), que serían los siguientes casos:

- Si la información no ha sido enviada previamente a dicho dispositivo, se envía.

- Si la información ha sido enviada, pero la transferencia no fue exitosa y la cantidad de intentos (*tries_on_failure*) no superó el umbral configurado para ese objeto o dato, se envía.

- Si la información ha sido enviada previamente con éxito, pero la cantidad de días que transcurrieron desde su último envío es mayor a la configurada en el dato (*days_to_repeat*), se envía.

- Estos casos se combinan con un filtrado previo, primero si el dato está habilitado (*published*) y luego si la fecha actual está entre el inicio (*start_publish_date*) y el fin (*end_publish_date*) de la publicación- .

Cualquier otra combinación producida, el dato no se envía.

Para una mejor interpretación, a continuación se muestra una figura con el flujo que se realiza en el envío a un dispositivo:

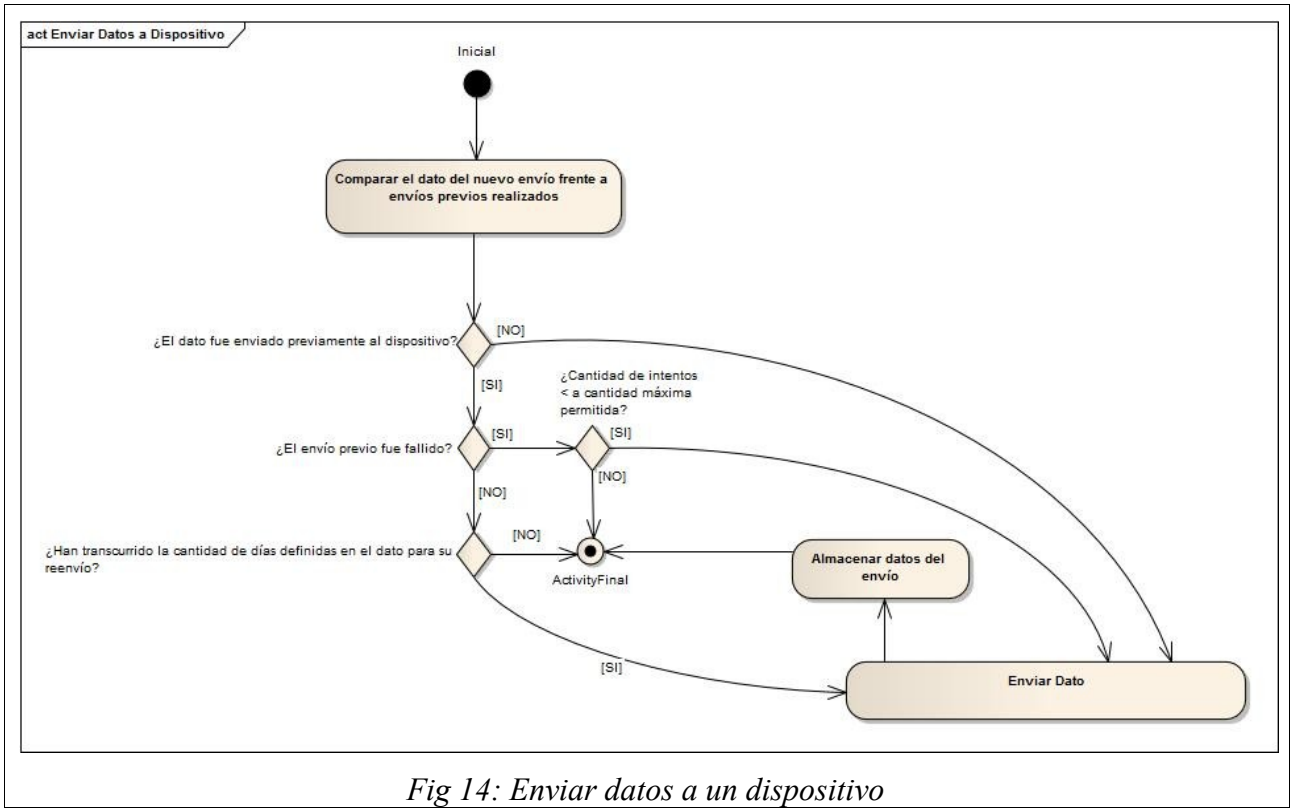


Fig 14: Enviar datos a un dispositivo

El chequeo anterior se realiza por cada dispositivo descubierto y, a su vez, por cada dato que haya sido cargado para el envío en la zona detectada.

4.5.2.2 Políticas de broadcasting (Blacklist o Whitelist)

Para este tipo de sistemas, resulta imprescindible contar con políticas de emisión de mensajes para que el administrador del mismo tenga la posibilidad de elegir cuál de las dos es la más adecuada para su recinto. Con estas políticas uno decide el perfil de emisión que desea para su entorno.

Por otro lado, también se le provee un mecanismo de suscripción y bloqueo al propio dispositivo de una manera totalmente trivial y sencilla para el usuario del dispositivo, y que se explicará más adelante.

El sistema posee las dos políticas más conocidas como *listas negras* (**blacklist**) y *listas blancas* (**whitelist**). Mediante la configuración del módulo Web uno puede seleccionar con qué política desea que el Sistema Ubicuo opere.

En cualquiera de las dos políticas que se usen, si bien hay casos que no corresponde, por más que no se envíe la información, todos los dispositivos detectados

serán registrados (si corresponde) de manera estadística sin tener esto ningún tipo de perjuicio ni interacción al circundante.

4.5.2.2.1 Blacklist

Si se elige una configuración de lista negra (blacklist), por defecto el sistema va a intentar realizar el envío a todos los dispositivos que se encuentren siempre y cuando no estén en dicha lista.

4.5.2.2.2 Whitelist

De manera contraria que con las listas negras, en este método se enviará información a aquellos dispositivos que se encuentren en la lista blanca, de manera contraria no se enviará ningún tipo de información.

4.5.2.2.3 Mecanismo de suscripción

Esta es una de las cualidades más importantes y mejor logradas en cuanto a las políticas de emisión, que es la forma en que los usuarios del sistema por sí solos pueden agregarse en la lista negra o blanca, dependiendo del caso, como así también salir de dichas listas.

Se pensaron diversas alternativas para poder salvar estas operaciones, que se enumeran a continuación:

- Una interface Web pública (sin necesidad de contar con credenciales para el acceso) en donde cada interesado ingresará la dirección MAC de su dispositivo interesado a agregar en alguna de las dos listas. Esta idea fue rápidamente descartada ya que en muchos dispositivos modernos no es fácil obtener dicha dirección desde el menú tradicional y, dependiendo la marca, existen diferentes formas de hacerlo, pero por lo general es agregando un código como por ejemplo *#2820# . Esto no es del todo trivial además de que también el interesado debería acceder a una página Web para suscribirse o desuscribirse a la emisión de la información.
- Debido a la complejidad de la obtención de la MAC por parte del usuario se pensó una alternativa en la que el interesado ingresa (en un formulario Web) el “nombre a

mostrar” de su teléfono. Esto no resultó una buena idea, ya que por defecto los celulares vienen con el modelo en dicho nombre y se estaría filtrando (o desfiltrando) otros celulares que tal vez sí deseaban (o no) recibir la información. A esta idea también se le suma el acceso a la página Web generando un malestar en aquellas personas que no están interesadas en recibir la información.

Por último se logró llegar a la solución finalmente implementada que es la siguiente:

- Desde la administración se configuran dos nombres clave (tabla *genericConfig*); uno para suscribirse al sistema (allow_name) y otro para desuscribirse (disallow_name), es decir, no recibir más mensajes. Una vez configurados estos nombres (por defecto “enviarme” y “noenviarme” respectivamente) el usuario simplemente debe ingresar en su nombre a mostrar lo que desea; dependiendo de la política configurada el sistema va a responder de diferente manera:

  Política Blacklist:

  - **Suscripción:** En esta política si el usuario configura en su dispositivo “noenviarme”, el sistema automáticamente lo agrega en la lista negra, le envía un mensaje de confirmación y nunca más se le va a enviar información a ese dispositivo hasta que el mismo no sea removido de dicha lista.
  - **Desuscripción:** De la misma manera que se utilizó para bloquear los mensajes a dicho dispositivo, se puede eliminar el mismo de la lista negra ingresando en su nombre la palabra “enviarme”; cuando sea detectado por alguno de los Discovers se le cambiará su estado y se enviará el mensaje de confirmación y la información referente a su zona.

  Política Whitelist:

  - **Suscripción:** Por defecto los celulares detectados no recibirán información hasta que no estén en dicha lista, por lo cual ingresando el nombre “enviarme”, al ser detectado, el sistema automáticamente lo agregará a la lista y le enviará información desde ese momento en adelante o hasta que se desuscriba.
  - **Desuscripción:** Si no se desea recibir información solamente con cambiar el nombre por “noenviarme” y ser alcanzado por algún Discover es

suficiente.

Cabe aclarar que en todos los casos el nombre deberá ser cambiado hasta que sea alcanzado por un Discover y llegue la confirmación, luego de eso puede volver a escribir el nombre que desee el usuario.

Cada vez que hay un cambio de estado, se envía un mensaje de confirmación con una imagen preestablecida que puede ser cambiada desde la administración remota. En caso de no haber ninguna cargada previamente, el sistema tomará los mensajes (en texto) de la tabla genericConfig (disallow_message y allow_message), lo transformará a imagen y lo enviará (previo almacenamiento local para evitar transformaciones indeseadas, se realiza una sola vez). Se trabajan con imágenes por el hecho de que son un estándar aceptado por todos los celulares y el protocolo de intercambio de objetos. Nos aseguramos de que el mensaje sea leído por todos los celulares ya que el manejo de texto (.txt) de cada marca y modelo difieren considerablemente. Con este mecanismo, además de ser sumamente cómodo para el usuario, es de fácil utilización; solo debe cambiar momentáneamente el nombre de su dispositivo.

Nótese que los nombres “enviarme” y “noenviarme” son a modo de ejemplo, pueden ser configurados desde la Administración remota como se desee, y una buena idea en el caso de usar Whitelist sería setearlo con una especie de contraseña no trivial para que no cualquiera se pueda registrar.

4.5.2.3 Modo debugging

Se han implementado 3 diferentes niveles de debugging por cada estación huésped con la finalidad de obtener un mayor o menor nivel de detalle de lo que está ocurriendo en tiempo real en la estación, sensores y las diferentes ubicaciones que se quieren monitorear.

- El nivel 0 (cero) da un mínimo de información y sólo muestra mensajes realmente críticos como ser error en la conexión a la base de datos, etc.
- El nivel 1 (uno) muestra un mayor nivel de detalle de lo que va ocurriendo, los dispositivos detectados, los servicios que cada uno de estos tiene publicados al igual que su dirección MAC y nombre.

- El nivel 2 (dos) muestra todos los mismos mensajes que el nivel 1 sumándole todas las transacciones contra la base de datos para poder tener un mejor y más completo panorama de todo lo que sucede en el sistema.

Dicha configuración del nivel de debugging elegido es propio de la estación por lo tanto posee un parámetro en su archivo de configuración local (`debug_mode_level`).

4.5.2.4 Detalles de la implementación (detalles técnicos y definición de clases)

A continuación, en la Fig. 15, se detallan las principales clases y cómo es la interacción entre ellas; las mismas son las más relevantes en cuanto al propósito general del módulo. Existen algunas clases más del modelo destinadas a mejorar la performance y al manejo de hilos de dicho módulo que no serán incluidas en este modelo para una mayor comprensión del núcleo del sistema.

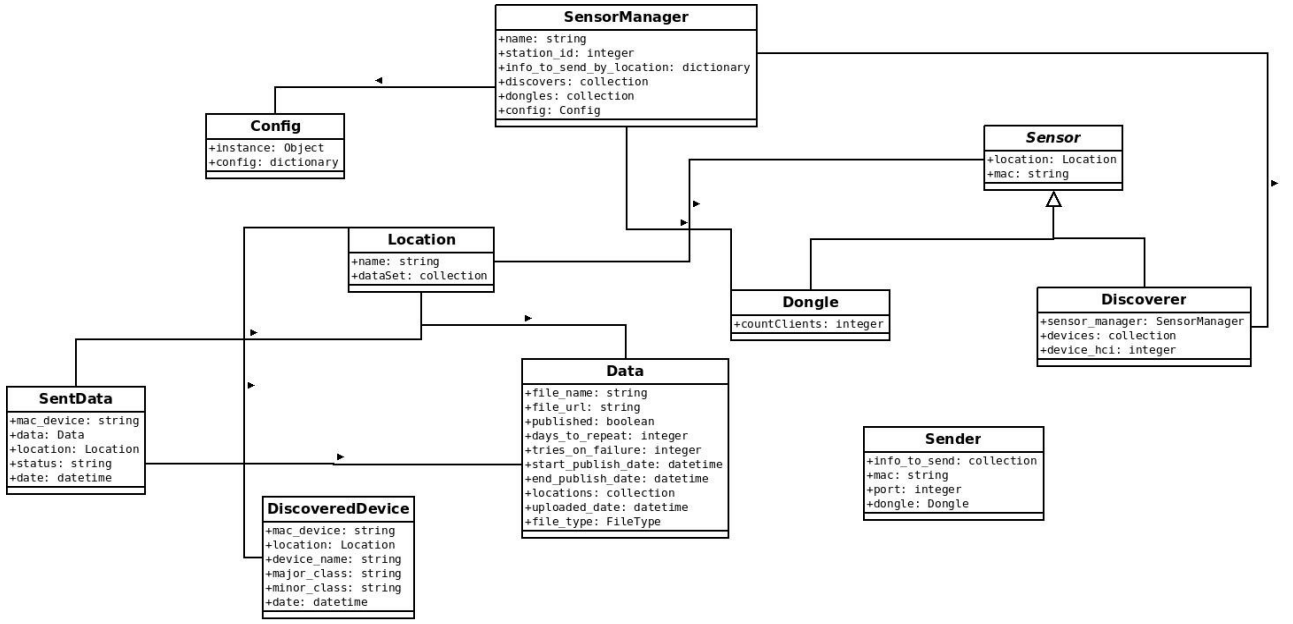


Fig 15: Diagrama de clases del Sistema Ubicuo

Descripción de las clases:

- Existe un pequeño módulo cargador (loader) que es el encargado de la puesta inicial del sistema. En una primer instancia, mediante la clase **Config**, obtiene los

parámetros del archivo de configuración, creando a su vez la conexión a la base de datos, y obtiene mediante el *sensor_id* del archivo de configuración la única instancia del **SensorManager** para esa estación. El archivo de texto plano con la configuración local puede ser como el siguiente (contiene datos a modo de ejemplo):

```
# Main DataBase Configuration
driverDataBase=mysql
userDataBase=root
passwdDataBase=mypass
IPDataBase=localhost
DataBaseName=blueps
# ID for the station host of dongles
station_id=1

# Main web server configuration
# example: http://192.168.1.1/blue4aps/files/ or
#http://mydomain.org/media
#····Deprecated····#
#server_address=http://127.0.0.1/

#OBEX Configuration
OBEX_OK=160

# Time Interval for the storage of the discovered #devices (in
minutes)
#····Deprecated····#
#time_discovers_storage=5
```

- Descripción:

Este archivo contiene los parámetros iniciales para la puesta en marcha del sistema, como ser accesos a la base de datos y el *station_id*. Este último debe coincidir con el configurado mediante el módulo Web para la estación en particular. Se puede observar también que existen dos parámetros que figuran obsoletos (deprecated); esto se debe a que fueron migrados a la Base de Datos (tabla *generic_config*) debido a una decisión de implementación. Esta decisión fue por una cuestión de una mayor accesibilidad a dichos parámetros ya que pueden ser

modificados mediante el módulo Web sin inconvenientes.

- **`Config`:**
  - Resumen: Clase encargada de la configuración local.
  - Variables de instancia:
    - `instance = None`
      Utilizado por el patrón Singleton para identificar a la instancia del objeto.
    - `config = {}`
      Una vez leído el archivo de configuración es en este diccionario donde guarda lo configurado en el archivo mencionado anteriormente.
  - Descripción:

    Implementa el patrón de diseño Singleton para mantener una única instancia de la clase y que pueda ser consultada por las clases que así lo requieran y cuantas veces lo requieran, pero el archivo de configuración mencionado anteriormente será leído una sola vez durante el ciclo de vida del objeto. Esto fue realizado de esta manera para evitar lecturas a disco innecesarias. Por otro lado, esta clase se encarga de obtener la configuración faltante que reside en la Base de Datos, añadiéndola a la obtenida desde el archivo de configuración (como ser *server_address* y *time_discovers_storage* mencionados más adelante).

- **`GenericConfig`:**
  - Resumen: Clase encargada de levantar de la base de datos los parámetros de configuración de la aplicación.
  - Variables de instancia:

- `server_address = StringCol()`
  Dirección URL del servidor donde se ejecuta la aplicación remota.

- `time_discovers_storage = IntCol()`
  Tiempo de espera en segundos para guardar los datos de un dispositivo encontrado.

- `disallow_name = StringCol()`
  Nombre que deberá ser escrito en el dispositivo para no recibir ningún dato en caso de que la política de envío de la aplicación sea de lista negra.

- `allow_name = StringCol()`
  Nombre que deberá ser escrito en el dispositivo para recibir datos en caso de que la política de envío de la aplicación sea de lista blanca.

- `disallow_message = StringCol()`
  Mensaje que será enviado al dispositivo luego que el mismo haya sido procesado para la aplcación de la política de envío de lista negra.

- `allow_message = StringCol()`
  Mensaje que será enviado al dispositivo luego que el mismo haya sido procesado para la aplicación de la política de envío de lista blanca.

- `disallow_image_url = StringCol()`
  Dirección URL de la imagen a enviar al dispositivo luego que el mismo haya sido procesado para la aplicación de la política de envío de lista negra.

- `allow_image_url = StringCol()`
  Dirección URL de la imagen a enviar al dispositivo luego que el mismo haya sido procesado para la aplicación de la política de envío de lista blanca.

- `policy_tx = IntCol()`
  Política de envío elegida para la aplicación

- `dongles_names = StringCol()`

Nombre asignado a los dongles asociados para el envío.

- Descripción:

  Representa al diccionario levantado de la base de datos por la clase Config para obtener los datos de configuración de la aplicación.

- **`SensorManager`:**

  - Resumen: Representa a la Estación portadora de los sensores, responsable de la transmisión y recolección de información en las distintas ubicaciones.

  - Variables de instancia:

    - `name = StringCol()`
      Hace referencia al nombre de la estación. Ejemplo: “Hall Central”

    - `station_id = IntCol()`
      Es el identificador de la estación, el mismo debe ser único en todo el sistema y debe coincidir con el mismo atributo de la configuración local del módulo (archivo *config* administrado por la clase **Config**)

    - `dongles = MultipleJoin("Dongle")`
      Este es el pool de dongles o sensores que tiene asociado esta estación. Cada uno de los cuales está asociada a una ubicación.

    - `discovers = MultipleJoin("Discover")`
      Este es el pool de descubridores (sensores destinados al descubrimiento de dispositivos) que tiene asociado esta estación. Cabe aclarar que una estación puede abarcar muchas ubicaciones y por cada una puede tener uno o varios descubridores.

  - Descripción:

    Esta clase es la que administra todos los sensores y los datos a transmitir.

Cuando el sistema se inicia, los descubridores asociados al **SensorManager** comienzan su labor (la de encontrar dispositivos). A medida que va descubriendo dispositivos, haciendo uso de otras clases (**MyDiscoverer** y **AsynchronousSend**), mediante la utilización de hilos, se le envía concurrentemente la información encontrada al **SensorManager**, quien delega a la clase **DiscoveredManager**, que decidirá si dicha información debe ser o no persistida (explicado en la clase DiscoveredDevice).

En términos generales esta clase es la encargada del manejo y recepción de sucesos y la posterior delegación a la clase correspondiente.

- **`Dongle` (Sensor)**

  - Resumen: Esta clase extiende de **Sensor** heredando las características comunes a los sensores, ya sean estos "descubridores/recolectores" (se los llama recolectores ya que al descubrir un dispositivo y sus características envía dicha información al **SensorManager** para su posterior almacenamiento) o "transmisores".

  - Variables de instancia:

    - `sensor_manager = ForeignKey("SensorManager")`
      Mantiene la relación con el **SensorManager** al cual pertenece.

    - `mac = StringCol()`
      Cada sensor tiene asociada su dirección MAC, la cual es utilizada, además de identificar los sensores, para abrir distintos sockets en la estación para obtener una mayor concurrencia.

    - `location = ForeignKey("Location")`
      Cada dongle o sensor está asociado a una ubicación.

    - `enabled = BoolCol()`
      Cada dongle posee un valor de "habilitado" para el uso de dicho sensor durante el proceso de envío.

- Descripción: Cada instancia sabe cuántas conexiones o clientes tiene en un momento determinado en la ubicación en donde se encuentra. Dicha información es clave para el balanceo de carga implementado por el **SensorManager**, evitando de esta manera la sobrecarga de algunos cuando otros están ociosos. Una vez determinado quién es el sensor con menor carga en ese momento, el **SensorManager** se lo envía al **Sender** para que comience con el envío a través de dicho dongle o sensor.

- **`Discoverer` (Sensor)**

  - Resumen: Representa el sensor dedicado al descubrimiento de dispositivos.

  - Variables de instancia:

    - `sensor_manager = ForeignKey("SensorManager")`
      Mantiene la relacion con el **SensorManager** al cual pertenece.

    - `mac = StringCol()`
      Cada sensor tiene asociada su dirección MAC, la cual es utilizada, además de identificar a los sensores, para abrir distintos sockets en la estación para obtener una mayor concurrencia.

    - `location = ForeignKey("Location")`
      Cada dongle o sensor está asociado a una ubicación.

  Descripción: Al igual que la clase **Dongle**, extiende de **Sensor** heredando el conocimiento expresado previamente, esta clase es la encargada de descubrir constantemente dispositivos. Una vez que el **SensorManager** obtiene todos sus descubridores, se inicia la ejecución concurrente de los mismos. Esta clase es la encargada de inicializar la búsqueda propiamente dicha y de procesar los resultados parciales. La búsqueda está representada por la clase **MyDiscoverer**. Se inicia el ciclo de búsqueda, cuando dicho ciclo termina, se

vuelve a comenzar. Una vez iniciado el proceso de detección por cada dispositivo, esta clase se encarga de verificar cuáles son los servicios que posee cada uno de los dispositivos encontrados, devolviendo sólo aquellos que cuentan con el servicio “Object Push” (`bluetooth.OBEX_OBJPUSH_CLASS`), ya que este es el utilizado por el Sistema Ubicuo para la transmisión de la información. Cabe aclarar que por cada dispositivo descubierto se hace un procesamiento paralelo por parte de la clase **MyDiscoverer**.

- **`MyDiscoverer` (`bluetooth.DeviceDiscoverer`)**
  - Resumen: Esta clase representa de alguna manera a la búsqueda en sí, procesando a cada dispositivo encontrado.
  - Variables de instancia:
    - `sock = None`
      Representa al socket abierto sobre el dispositivo indicado en *device_id.*
    - `is_inquiring = False`
      Indica si está o no en proceso de indagación.
    - `device_id`
      Es el número de HCI del dongle que se ha configurado como descubridor. Se utiliza para crear el socket sobre el dongle.
    - `location`
      Es la ubicación sobre la cual está indagando.
    - `devices = []`
      Se guardan todos los dispositivos a medida que se van encontrando.
  - Descripción: Esta clase es creada por el **Discover** por cada ciclo de búsqueda. En cada ciclo puede o no detectar dispositivos; si lo hace, se maneja cada

descubrimiento con el método `def` **`device_discovered`**`(self, address, device_class, name):` en donde luego se ubica al dispositivo descubierto en la clasificación correspondiente a los bits del *device_class* según la tabla publicada por Bluetooth[17]. La misma tiene una clasificación más general (*major_classes*) y dependiendo de cual sea la correspondiente, va a tener una clasificación diferente (*minor_classes*). El campo de clase de Dispositivo/Servicio se divide de la siguiente forma:

Del bit 23 al 13	***Del bit 12 al 8***	***Del bit 7 al 2***	***Del bit 0 al 1***
Reservado para las clases de servicios	*Major Device Class*	*Minor Device Class*	*00 (Tipo de formato)*

Tabla 5: Especificación del campo de clase de Dispositivo/Servicio

Major device Class	***bits***				
	12	*11*	*10*	*9*	*8*
Miscellaneous	*0*	*0*	*0*	*0*	*0*
Computer	*0*	*0*	*0*	*0*	*1*
Phone	*0*	*0*	*0*	*1*	*0*
LAN/Network Access point	*0*	*0*	*0*	*1*	*1*
Audio/Video	*0*	*0*	*1*	*0*	*0*
Peripheral	*0*	*0*	*1*	*0*	*1*
Imaging	*0*	*0*	*1*	*1*	*0*
Uncategorized, specific device code not specified	*1*	*1*	*1*	*1*	*1*
All other values reserved	*X*	*X*	*X*	*X*	*X*

Tabla 6: Clasificación de las clases de dispositivos según los bits.

17 https://www.bluetooth.org/Technical/AssignedNumbers/baseband.htm

El sistema va a subclasificar a los dispositivos dependientes de Computer o Phone de la siguiente manera:

Minor device Class (Computer Major)	***bits***					
	7	*6*	*5*	*4*	*3*	*2*
Uncategorized, code for device not assigned	*0*	*0*	*0*	*0*	*0*	*0*
Desktop workstation	*0*	*0*	*0*	*0*	*0*	*1*
Server-class computer	*0*	*0*	*0*	*0*	*1*	*0*
Laptop	*0*	*0*	*0*	*0*	*1*	*1*
Handheld PC/PDA (clam shell)	*0*	*0*	*0*	*1*	*0*	*0*
Palm sized PC/PDA	*0*	*0*	*0*	*1*	*0*	*1*
Wearable computer (Watch sized)	*0*	*0*	*0*	*1*	*1*	*0*
All other values reserved	*X*	*X*	*X*	*X*	*X*	*X*

Tabla 7: Subclasificación de dispositivos de la clase Computer

Minor device Class (Phone Major)	*bits*					
	7	*6*	*5*	*4*	*3*	*2*
Uncategorized, code for device not assigned	*0*	*0*	*0*	*0*	*0*	*0*
Cellular	*0*	*0*	*0*	*0*	*0*	*1*
Cordless	*0*	*0*	*0*	*0*	*1*	*0*
Smart phone	*0*	*0*	*0*	*0*	*1*	*1*
Wired modem or voice gateway	*0*	*0*	*0*	*1*	*0*	*0*
Common ISDN Access	*0*	*0*	*0*	*1*	*0*	*1*
All other values reserved	*X*	*X*	*X*	*X*	*X*	*X*

Tabla 8: Subclasificación de dispositivos de la clase Phone

La información de catalogación del dispositivo está en el campo Clase de Dispositivo/Servicio, representado en *device_class* indicado en el Assigned Numbers - Bluetooth Baseband, el cual contiene la información tanto del tipo de dispositivo como también del tipo de servicio brindado como se ha mencionado.

Para obtener el tipo de dispositivo se realizan operaciones de corrimiento de bits y ANDs lógicos:

```
major_class = (device_class >> 8) & 0xf
```

Primero se corren 8 bits (del 0 al 7 ya que indican el tipo de formato y el minor device class) y luego se realiza la operación AND lógica con F (Hexa) = 11111 (binario) para quedarnos con el major device class.

Para obtener el minor device class se realiza una operación similar:

```
minor_class = (device_class >> 2) & 0x3f
```

Se desechan los dos primeros bits (tipo de formato) y luego se realiza el AND lógico con 3F (Hexa) = 111111 (binario) para quedarnos con los bits que hacen referencia

al minor device class (nótese que con los 3 primeros bits bastaría pero se completa por una eventual y futura expansión)

Luego, dependiendo de qué Major Device Class sea el dispositivo, se va a identificar el Minor Device Class.

De esta Sub Clase de la librería utilizada se reimplementaron los métodos:

```
def device_discovered(self, address, device_class, name)
```

y

```
def __init__ (self, discover,location, device_id=-1)
```

Este último se reimplementó para poder enviarle el *device_id* que se corresponde con el número de HCI del sensor. Por lo tanto se va a realizar la búsqueda con el sensor que así fuera definido desde la configuración para realizar dicha tarea.

El método *device_discover* es invocado cada vez que se encuentra un dispositivo y luego de catalogarlo como se explicó previamente, de manera concurrente se envían los datos recabados a **AsynchronousSend** que es el encargado del diálogo con el **SensorManager** para persistir o no dicha información (ver la clase **DiscoveredDevice** en donde explica si se guardan o no los datos).

- **`DiscoveredDevice`**
  - Resumen: Representa cada descubrimiento a almacenar
  - Variables de instancia:
    - `mac_device = StringCol()`
      Dirección MAC del dispositivo del cual se ha transmitido.
    - `device_name = StringCol()`
      Es el nombre del dispositivo encontrado (el nombre con el que se muestra)
    - `location_id = IntCol()`

Ubicación en la cual fue descubierto el dispositivo y por ende de la cual se va a transmitir.

- `major_class = StringCol()`
  Clasificación general del dispositivo encontrado, es la rama principal en la cual cae el mismo (explicado en detalle en la clase MyDiscoverer).

- `minor_class = StringCol()`
  Es la clasificacion del dispositivo dentro de una categoria mayor (major_class) explicado en detalle en la clase MyDiscoverer.

- `date = DateTimeCol()`
  Es la fecha y hora en que fue encontrado el dispositivo.

- **Descripción:** Esta clase representa al dispositivo encontrado que se va a almacenar en la base de datos con la información más relevante a fines estadísticos. Previo al almacenamiento se realiza una consulta para verificar si dicho dispositivo ya fue almacenado previamente en la misma ubicación y no han transcurrido la cantidad de minutos suficientes dados por *Config.time_discovers_storage. En ese caso,* no se realiza el almacenamiento; de cualquier otra manera, se almacena el dispositivo encontrado con toda la información antes mencionada.

- **Sender(`threading.Thread`)**

  - Resumen: Representa la comunicación con los dispositivos.

  - Variables de instancia:

    - `sensorManager = SensorManager`

      `infoToSend:` Es una colección de Data en donde cada instancia tiene todo lo necesario referente al dato a enviar (ver definición de la clase **Data**).

      `Dongle:` Conoce al sensor por el cual realizar la transición.

`MAC:` Dirección MAC del dispositivo al cual va enviar la información.

`port:` Puerto al cual deberá establecer la conexión.

- Descripción: Esta clase es creada por el **SensorManager** quien le provee lo necesario para el envío de información a un dispositivo. Es una subclase de **Threading.Thread** para proveer su capacidad de concurrencia y así evitar que otros envíos sean demorados (concurrencia). Dependiendo de la ubicación en la cual fue detectado el dispositivo, el **SensorManager** busca el dongle menos ocupado y la información relacionada con esa zona y se lo envía al **Sender**. Dicha instancia abre un socket hacia el dispositivo (mediante la combinación de MAC y port) y uno a uno va enviando todos los datos pertinentes. Previo al envío, por cada dato a mandar se realiza un chequeo si corresponde o no enviarlo mediante la invocación del método estático *sent_data.SentData.haveToSend (info,self.MAC)* a la clase **SentData** con la instancia del dato y la MAC del cliente (explicado en 4.5.2.1).

- **`SentData`**

  - Resumen: Realiza el control de la información enviada a un dispositivo en particular.

  - Variables de instancia:

    - `mac_device = StringCol()`
      Dirección MAC del dispositivo del cual se ha transmitido.

    - `data_id = IntCol()`
      La referencia del dato enviado.

    - `location_id = IntCol()`
      Ubicación en la cual fue descubierto el dispositivo y por ende de la cual se va a transmitir.

    - `status = StringCol()`

Es el estado de las transmisiones realizadas previamente (Success o Reject).

- `date = DateTimeCol()`
  Fecha y hora en la cual se realizó una determinada transmisión.

- Descripción: Mediante el método estático *@staticmethod def* ***haveToSend*** *(data,mac_address):* Esta clase realiza las consultas a la base de datos y evalúa si se debe enviar un dato a un dispositivo en particular devolviendo al **Sender** si ese dato a ese dispositivo tiene que enviarse o no. Las condiciones de envío son las enunciadas en 4.5.2.1

- **Data**

  - Resumen: Representa al dato a enviar o enviado.

  - Variables de instancia:

    - `locations = RelatedJoin("Location")`
      Referencia a las ubicaciones asociadas al dato.

    - `file_name = StringCol()`
      Nombre utilizado para la transmisión (podría ser distinto al nombre físico).

    - `file_url = StringCol()`
      URL en donde reside el archivo a partir del árbol Web (document root). Ejemplo: *"media/salidas_de_emergencia.mp3"*

    - `published = IntCol()`
      Referencia si el dato está o no habilitado para la transmisión.

    - `days_to_repeat = IntCol()`
      Cantidad de días que deben transcurrir para un reenvío del mismo dato previamente enviado al dispositivo. Es 0 (cero) para un único envío.

- `tries_on_failure = IntCol()`
  Cantidad de intentos ante una eventual falla a un mismo dispositivo. Una vez superado dicho límite, el sistema cesa la transmisión.

- `start_publish_date = DateTimeCol()`
  Fecha y hora de inicio de la publicación del dato.

- `end_publish_date = DateTimeCol()`
  Fecha y hora de fin de la publicación del dato.

- Descripción: Cuando todas las instancias son obtenidas de la base de datos, éstas no tienen asociado el dato en binario para la transmisión. La primera vez que sea necesaria la transmisión, esta clase se encarga de obtener su propio dato en caso que corresponda el envío. Es decir, que cada vez que se necesite enviarlo, se invoca al método *def **getData**(self)*: en donde se verifica que tenga asociado el dato, si no lo tiene, se realizan todos los chequeos previos para no tener instanciada información que no va a ser transmitida nunca, porque está deshabilitada (published) o porque ya venció la fecha de publicación, se verifica de la siguiente manera:

```
if self.published == 1 and (self.start_publish_date <= today and today
<= self.end_publish_date)
```

Es decir, si está publicado y la fecha de publicación está en vigencia, se obtiene el dato en binario mediante el protocolo HTTP al servidor indicado en la base de datos (*generic_config*). Las siguientes invocaciones ya van a tener su dato asociado, por lo que no es necesario una nueva búsqueda y obtención del dato desde el servidor que lo aloja (Lazy Evaluation). Esto fue realizado de esta manera por cuestiones netamente de performance y eficiencia, para evitar que los objetos ocupen lugar en algo que nunca será referenciado y transmisiones por la red innecesarias. Dependiendo de qué tipo de objeto sea, su tamaño podría ser importante (mp3, jpg, etc).

4.5.2.4.1 Modificaciones sobre la librería PyBluez (bluez.py)

Para poder realizar este proyecto de acuerdo a las necesidades planteadas, fue necesario realizar algunas modificaciones sobre la librería PyBluez. Se agregaron tres funciones dentro de la misma para obtener una mejor performance, una mayor concurrencia y flexibilidad.

A continuación se detallan los métodos implementados y su funcionalidad:

- `def` **`read_local_bdaddr`**`(hci_sock):`

  Esta función recibe el socket establecido de un dongle y devuelve la dirección MAC del dongle que posee dicho socket.

- `def` **`MACcompletion`**`(x):`

  Simplemente recibe la dirección MAC y la completa con un 0 (cero) adelante de cada uno de los dígitos hexadecimales (si es que faltaba). Dicha función fue requerida ya que en el proceso de obtener la MAC de un dongle al tratar cada parte como números enteros, si la misma empezaba con 0, éste era removido y luego no concordaba con la dirección del dongle. Por ejemplo, la dirección obtenida del HCI 1 es 0:15:3:15:A2:28. Luego, la función devuelve 00:15:03:15:A2:28.

- `def` **`getHCIFromMAC`**`(mac):`

  Esta función recibe una dirección MAC y busca sobre cada socket creado localmente, invocando a las funciones anteriormente mencionadas, y da como resultado el número de HCI que corresponde a esa dirección MAC. Dicho número o identificador es otorgado por el sistema operativo dependiendo de cómo se inician los dongles.

  - Caso 1

    Devices:

    hci0 00:15:83:15:A3:10

hci1 00:15:83:15:A2:28

- Caso 2

Devices:

hci0 00:15:83:15:A2:28

hci1 00:15:83:15:A3:10

El inconveniente que surgió se debe a que la librería utiliza el número de HCI para iniciar el descubrimiento sobre un dongle; y como este número es asignado de diferentes maneras y depende de varios factores del SO, es que se debieron realizar dichas modificaciones sobre la librería. Con ellas se logró que desde la Administración Remota con tal solo colocar la dirección MAC deseada para realizar el descubrimiento, podamos obtener qué HCI tiene dicha MAC y así poder iniciar el descubrimiento con el sensor que así fue designado independientemente de qué HCI le haya asignado el Sistema Operativo.

4.5.3 Administración Remota

4.5.3.1 Introducción

El sistema se puede administrar remotamente, pudiendo acceder al mismo desde cualquier computadora con acceso a la red, ya sea desde intranet o internet, y en el último caso, permitiendo configurar, acceder y monitorear el sistema desde cualquier punto conectado a internet.

Este módulo cuenta con un acceso seguro (mediante usuario y contraseña), diferentes perfiles de usuario (Administrador, Operador y Consulta), administración de los sensores y estaciones, acceso al módulo de estadísticas, monitoreo en tiempo real de los dispositivos encontrados y datos enviados, etc.

En esta sección se mencionan y explican las clases más relevantes del modelo del módulo Web. De las clases planteadas sólo se muestran el conocimiento y comportamiento más destacado para la comprensión del mismo.

4.5.3.2 Funcionamiento

Mediante una URL en cualquier navegador se puede acceder a la Administración Remota del sistema. La misma se encuentra protegida con usuario y contraseña, para la cual se definieron distintos perfiles de usuario: administrador, operador y consulta. Dependiendo de ellos, el usuario tendrá distintas operaciones permitidas dentro del módulo.

Una vez dentro de la Administración, se pueden agregar Administradores de Sensores (*dongle stations)*, Descubridores, Dongles y Datos, los cuales son utilizados por el Sistema Ubicuo para la transferencia de información hacia los dispositivos encontrados.

Un **usuario de Consulta** puede realizar las siguientes operaciones:

- Ver los Lugares existentes del recinto.
- Ver los Datos agregados para los envíos.
- Ver los Dispositivos encontrados.
- Ver los Datos enviados a los dispositivos.

Un **Operador**, además de realizar las operaciones anteriores, también puede:

- Agregar, modificar y eliminar los Lugares del recinto.
- Agregar modificar y eliminar los Datos para los envíos.
- Agregar, modificar y eliminar los Sensores asociados a las máquinas huésped.
- Agregar, modificar y eliminar los dongles Descubridores y Emisores asociados a cada Sensor y Lugar.
- Agregar, modificar y eliminar Tipos de Archivos para los Datos a enviar.
- Agregar, modificar y eliminar Plantillas para los distintos Tipos de Archivos de los Datos.

El **Administrador**, además de realizar las operaciones anteriores, también puede:

- Agregar, modificar y eliminar Usuarios.
- Agregar, modificar y eliminar Perfiles de Usuario (que son los tres mencionados).

- Modificar la configuración por defecto del sistema.
- Agregar, modificar y eliminar los dispositivos en las Listas Negra y Blanca.

La Administración de los perfiles de usuario y sus permisos se basa en las operaciones nombradas anteriormente, pero queda abierta a ser personalizada como se desee.

4.5.3.3 Configuración del sistema

Adicionalmente a la configuración de cada estación (mencionada en el Cap. 4.5.2.4), existen configuraciones generales que se aplican a todo el sistema, es decir que cada *dongle-station* toma dichos parámetros y los aplica a su funcionamiento. A continuación la Fig. 16 muestra las diferentes variables de configuración dentro de la Administración Remota:

Fig 16: Configuración general

Seguidamente se describe cada parámetro de configuración y su función dentro del sistema:

- **URL Servidor:** Este dato es utilizado por el *dongle-station* para saber de dónde obtener todos los recursos (imágenes, audio, contacto, evento, texto, etc.)
- **Tiempo de Espera de Almacenamiento:** Se configura un intervalo en minutos desde que un dispositivo fue almacenado hasta que se vuelve a almacenar (por dispositivo y por lugar). Por ejemplo, si el valor es de 10 y un dispositivo permanece al alcance de un sensor en una hora, solo será almacenado 6 veces. De no existir este mecanismo, se almacenaría 480 veces por el mismo sensor en la misma zona (existen 8 detecciones por minuto por cada sensor descubridor).

- **Nombre de Desuscripción:** Es utilizado por la política de broadcasting, en el cual se puede especificar el nombre requerido del dispositivo para desuscribirse del sistema. Según la Fig. 16, si un dispositivo ingresa “noenviar” en su nombre, el mismo no recibirá más información.
- **Nombre de Suscripción:** Al igual que el item anterior, es utilizado por la política de broadcasting para suscribir a un dispositivo al servicio. En la política de lista blanca es indispensable para comenzar a interactuar con el sistema.
- **URL Imagen de Desuscripción:** Se especifica la imagen que se envía a un dispositivo cuando éste decide desuscribirse del sistema, a modo de confirmación de la operación.
- **URL Imagen de Suscripción:** Análogamente al item anterior, se especifica la imagen a enviar cuando un dispositivo decide suscribirse al sistema.
- **Mensaje de Desuscripción:** Si no se estableció una imagen de desuscripción o ante una eventual falla en la recuperación del recurso, se toma el texto ingresado en este campo, se codifica en una imagen y se lo envía al dispositivo a modo de confirmación. Es un método alternativo a las imágenes.
- **Mensaje de Suscripción:** De la misma manera que el item anterior, se trata de un texto a ser codificado como imagen ante la falta o falla en la recuperación del recurso configurado en “URL Imagen de Suscripción”.

- **Política de Envío:** Se trata de la elección de la política de broadcasting (mencionada en el Cap. 4.5.2.2).
- **Nombre de los Sensores:** Es el nombre a mostrar utilizado por todos los sensores para la transmisión de la información. Dicho nombre es visualizado en los mensajes de confirmación de los dispositivos. Según la Fig. 16, en el dispositivo se mostrará "¿Desea recibir datos de FCs. JyS – UNLP?" al intentar realizar el envío.

4.5.3.4 Detalles de la implementación

Las tablas del modelo se corresponden, en su mayoría, a las clases ya descriptas en el Sistema Ubicuo. Sólo se agregaron algunas relacionadas a los datos que se van a crear para el envío hacia los dispositivos, como ser *file_type*, *template* y *template_field*, que más adelante se detallarán mejor.

El siguiente gráfico muestra las tablas de la base de datos, ya que este módulo administra básicamente toda la información almacenada en ella:

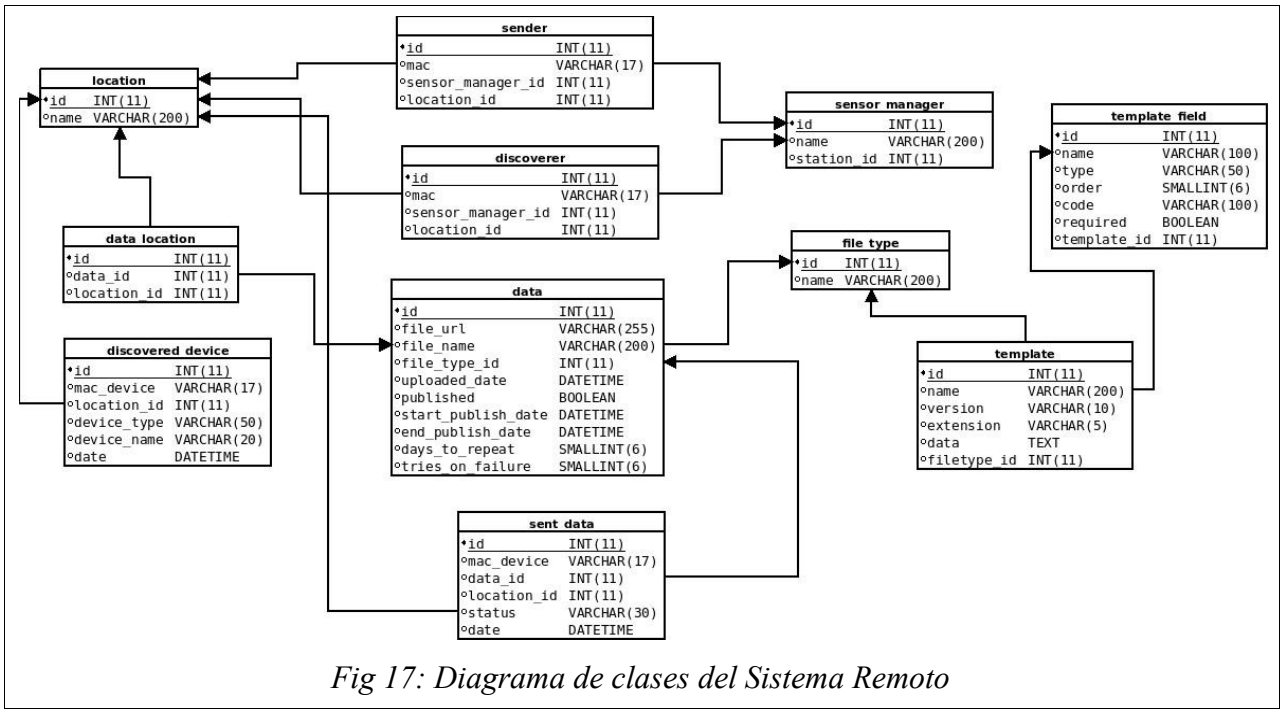


Fig 17: Diagrama de clases del Sistema Remoto

Una cuestión a tener en cuenta es que los Datos se encuentran implementados de forma separada de los lugares, sensores y dispositivos. Una vez que se agregan todos los **Lugares** del recinto en donde se desean ubicar los sensores Bluetooth y los

Administradores de Sensores (*dongle-station*) que representarán las estaciones huésped donde correrá el Sistema Ubicuo, se deben relacionar los mismos con los dongles **Emisores** y **Descubridores**.

Por otro lado, se agregarán los **Datos** que se deseen enviar, y que en principio pueden ser de varios tipos: imagen, texto, contacto, evento y audio. La acalarción “en principio” se debe a que el sistema queda abierto a la posibilidad de agregar más tipos, pero teniendo en cuenta que cada uno puede o no tener asociado un comportamiento diferente. Si bien un Dato para el sistema siempre se trata de un archivo para enviar a los dispositivos encontrados, dependiendo de su tipo, si el mismo se agrega como un archivo alojado en la máquina desde donde se esté ejecutando el módulo de Administración, o bien si el mismo se generará automáticamente basándose en una plantilla, como es el caso de los contactos, eventos y texto plano.

Una **Plantilla** es una base para un tipo de archivo determinado y que puede contener varios campos representando la información dentro de la estructura de la misma. El origen de la creación de las plantillas surge a raíz de los estándares existentes para el manejo de los contactos (vCard) y eventos (iCalendar) de los dispositivos móviles. Por ejemplo una plantilla del tipo vCard versión 3.0 puede ser similar al siguiente:

```
BEGIN:VCARD
VERSION:3.0
FN:%formatted_name%
N:%name%
NICKNAME:%nickname%
EMAIL;TYPE=INTERNET:%email%
TEL;TYPE=CELL:%cellphone%
TEL;TYPE=WORK:%work_phone%
TEL;TYPE=HOME:%home_phone%
ADR;TYPE=HOME:%address%
END:VCARD
```

Toda la información referente al contacto se maneja como variables encerradas entre porcentajes (%) y cada una se agrega como campo de la plantilla. Dichas variables deben tener como nombre el especificado en el código del campo, y cada uno de ellos podrá ser de varios tipos: texto, fecha, hora o número.

Además, cada campo posee un orden en el cual se mostrarán los mismos cuando sean completados al generar el archivo utilizando la plantilla seleccionada. Un campo puede ser o no obligatorio, y para el caso que lo sea, el archivo no podrá ser generado hasta que el usuario los complete.

Las plantillas pueden tener un número de versión, ya que existen diferentes versiones para cada estándar, y no todos los dispositivos móviles las interpretan. Otro dato asociado a las plantillas es su extensión, que se corresponde con la extensión del archivo que se generará a partir de la misma.

La plantilla para el tipo de archivo de texto es muy particular, porque sólo contiene una variable de tipo texto largo y que será reemplazada con todo el texto que se desee. Su extensión será del tipo .txt.

Un Dato, además del archivo, estará asociado a uno o varios lugares adonde se desee enviar junto con información de publicación, como ser:

- si el mismo se encuentra publicado o no
- fecha y hora de inicio de publicación
- fecha y hora de fin de publicación
- cantidad de días a repetir
- número de intentos ante falla

Todas estas propiedades se encuentran explicadas en la especificación del Sistema Ubicuo, pero básicamente un dato no se envía si no se encuentra publicado, y si tiene **fecha de inicio y fin**, se enviará sólo si la fecha de envío se encuentra entre las mismas. La **cantidad de días a repetir** se refiere a los días en que se reenviará el dato a un mismo dispositivo; por ejemplo, si la cantidad es 2 y ya se envió el dato en el día de hoy a un dispositivo determinado, el mismo no se volverá a enviar hasta pasado mañana. Y por último el **número de intentos ante falla** se refiere al número de veces que intentará retransmitirse el dato a un mismo dispositivo ante alguna falla en el envío.

4.5.3.4.1 Formato vCard:

Una vCard es un formato estándar de archivo utilizado para tarjetas electrónicas personales o de negocios, con información como el nombre, domicilio, teléfono, email, URLs, logos, fotografías e incluso clips de audio. Su objetivo principal es el de intercambiar información acerca de personas y recursos.

Dicha especificación se adapta como un formato intercambiable entre sistemas o aplicaciones, y su formato es definido de manera independiente al método utilizado para transportarlo, el cual puede ser a través de un archivo, de Internet, de redes inalámbricas o por código de barras bidimensionales; su extensión es .vcf o .vcard.

Para agregar una vCard en el sistema, se elige el tipo de archivo “Contacto” y una plantilla, lo que permite mostrar sus campos asociados para poder completarlos:

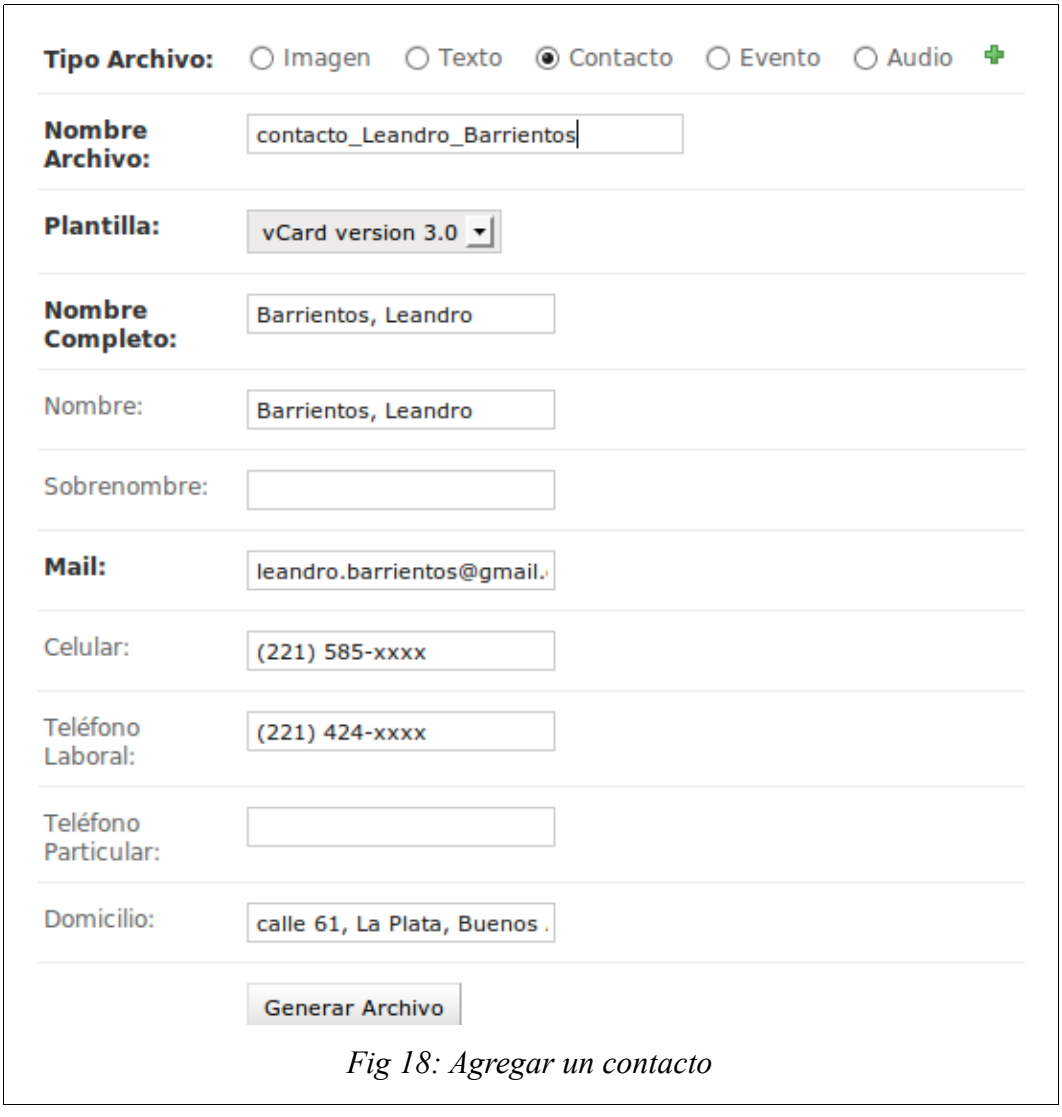


Fig 18: Agregar un contacto

Y de manera transparente al usuario, el sistema genera el siguiente archivo:

```
BEGIN:VCARD
VERSION:3.0
N:Barrientos, Leandro
FN:Leandro Barrientos
ORG:Facultad de Informática
TEL;TYPE=CELL:(221) 585-xxxx
TEL;TYPE=HOME:(221) 424-xxxx
ADR;TYPE=HOME:calle 61; La Plata; Buenos Aires; Argentina
EMAIL;TYPE=INTERNET:leandro.barrientos@gmail.com
END:VCARD
```

4.5.3.4.2 Formato iCalendar

Un iCalendar es un estándar para el intercambio de información de eventos y tareas. También se lo conoce como iCal, ya que éste es el nombre de la primer aplicación en implementarlo. Generalmente se utiliza para distribuir la información de un evento en particular, indicando hora, fecha y lugar, y su transmisión es independiente del protocolo de transporte. Su extensión es .ics y es soportado por varias aplicaciones como Google Calendar, Apple iCal y Yahoo Calendar, entre otras.

Para agregar un iCalendar se elige el tipo de dato “Evento” y una de sus plantillas para completar, al igual que en el vCard, sus campos asociados:

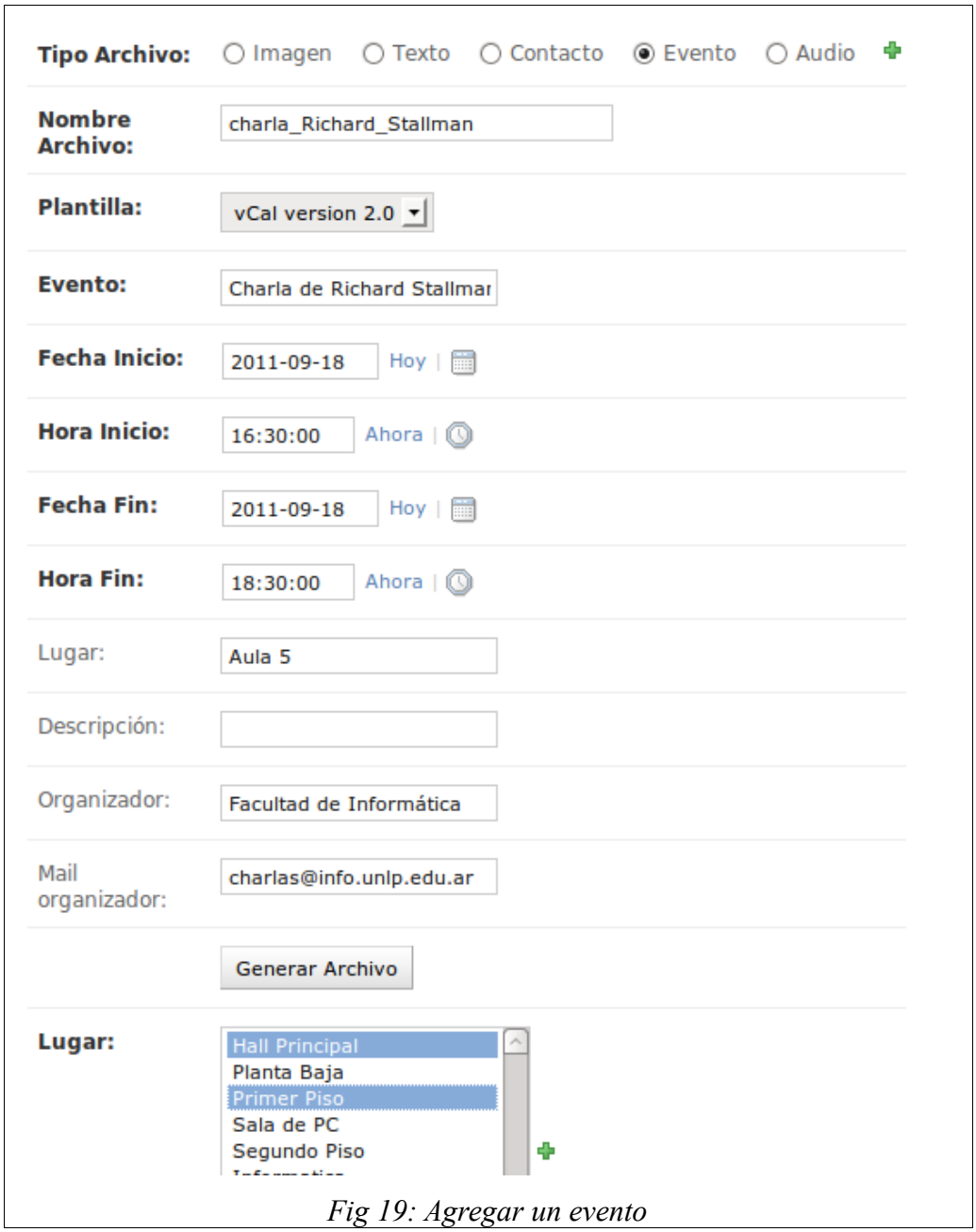


Fig 19: Agregar un evento

Luego, el archivo generado de manera transparente al usuario es el siguiente:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VEVENT
UID:uid1@example.com
DTSTAMP:20110910T170000Z
ORGANIZER;CN=:MAILTO:charlas@info.unlp.edu.ar
DTSTART:20110918T163000Z
```

```
DTEND:20110918T183000Z
SUMMARY:Charla de Richard Stallman
END:VEVENT
END:VCALENDAR
```

Si se desean agrupar varios eventos en un mismo archivo, la plantilla debería generarse con más de un par de sentencias `BEGIN:VEVENT` y `END:VEVENT`, dentro de las cuales se define un evento. También pueden incluir una alarma (`VALARM`) y pueden no tener fecha de comienzo (`DTSTART`) ni fecha de fin (`DTEND`), ya sea para cumpleaños o recordatorios.

Otra aplicación de este formato es para especificar tareas, como por ejemplo la siguiente:

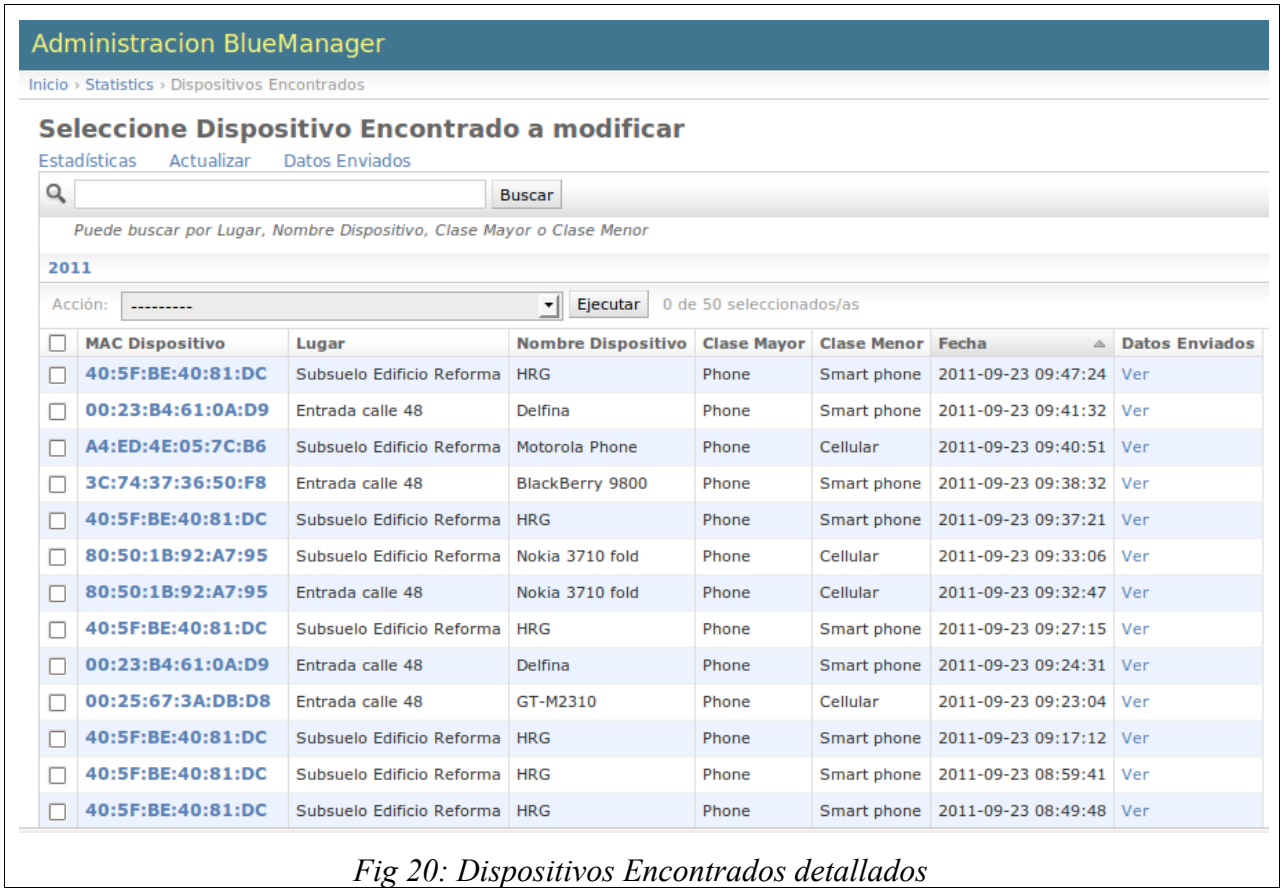
```
BEGIN:VTODO
UID:20110901T130000Z-123404@host.com
DTSTAMP:20110901T1300Z
DTSTART:20110415T133000Z
DUE:20110416T045959Z
SUMMARY:Preparar presentación de la clase
CLASS:CONFIDENTIAL
CATEGORIES:PERSONAL
PRIORITY:1
STATUS:NEEDS-ACTION
END:VTODO
```

A la tarea, al igual que a un evento, se le puede agregar una alarma y que la misma se repita una cantidad determinada de veces.

Dispositivos Encontrados

Continuando con el sistema de Administración Remota, como se muestra en la Fig. 19, en **Dispositivos Encontrados** se podrán ver todos los detalles de los dispositivos Bluetooth que fue encontrando el Sistema Ubicuo a lo largo del tiempo y según el parámetro de configuración *time_discovers_storage*, que especifica cada cuántos minutos deja de registrarse en la base un mismo dispositivo encontrado en un mismo lugar. De

cada dispositivo se podrá saber la fecha y hora que fue encontrado, en qué lugar, qué tipo de dispositivo es y su dirección MAC.



MAC Dispositivo	Lugar	Nombre Dispositivo	Clase Mayor	Clase Menor	Fecha	Datos Enviados
40:5F:BE:40:81:DC	Subsuelo Edificio Reforma	HRG	Phone	Smart phone	2011-09-23 09:47:24	Ver
00:23:B4:61:0A:D9	Entrada calle 48	Delfina	Phone	Smart phone	2011-09-23 09:41:32	Ver
A4:ED:4E:05:7C:B6	Subsuelo Edificio Reforma	Motorola Phone	Phone	Cellular	2011-09-23 09:40:51	Ver
3C:74:37:36:50:F8	Entrada calle 48	BlackBerry 9800	Phone	Smart phone	2011-09-23 09:38:32	Ver
40:5F:BE:40:81:DC	Subsuelo Edificio Reforma	HRG	Phone	Smart phone	2011-09-23 09:37:21	Ver
80:50:1B:92:A7:95	Subsuelo Edificio Reforma	Nokia 3710 fold	Phone	Cellular	2011-09-23 09:33:06	Ver
80:50:1B:92:A7:95	Entrada calle 48	Nokia 3710 fold	Phone	Cellular	2011-09-23 09:32:47	Ver
40:5F:BE:40:81:DC	Subsuelo Edificio Reforma	HRG	Phone	Smart phone	2011-09-23 09:27:15	Ver
00:23:B4:61:0A:D9	Entrada calle 48	Delfina	Phone	Smart phone	2011-09-23 09:24:31	Ver
00:25:67:3A:DB:D8	Entrada calle 48	GT-M2310	Phone	Cellular	2011-09-23 09:23:04	Ver
40:5F:BE:40:81:DC	Subsuelo Edificio Reforma	HRG	Phone	Smart phone	2011-09-23 09:17:12	Ver
40:5F:BE:40:81:DC	Subsuelo Edificio Reforma	HRG	Phone	Smart phone	2011-09-23 08:59:41	Ver
40:5F:BE:40:81:DC	Subsuelo Edificio Reforma	HRG	Phone	Smart phone	2011-09-23 08:49:48	Ver

Fig 20: Dispositivos Encontrados detallados

Además se podrán ver gráficos con estadísticas (los mismos fueron realizados en base a pruebas del prototipo con distintos dispositivos prestados).

- Por Fecha: Por defecto muestra todos los dispositivos encontrados en las últimas 4 semanas en todos los lugares del recinto, pero se puede personalizar la búsqueda entre dos fechas y para un lugar determinado.

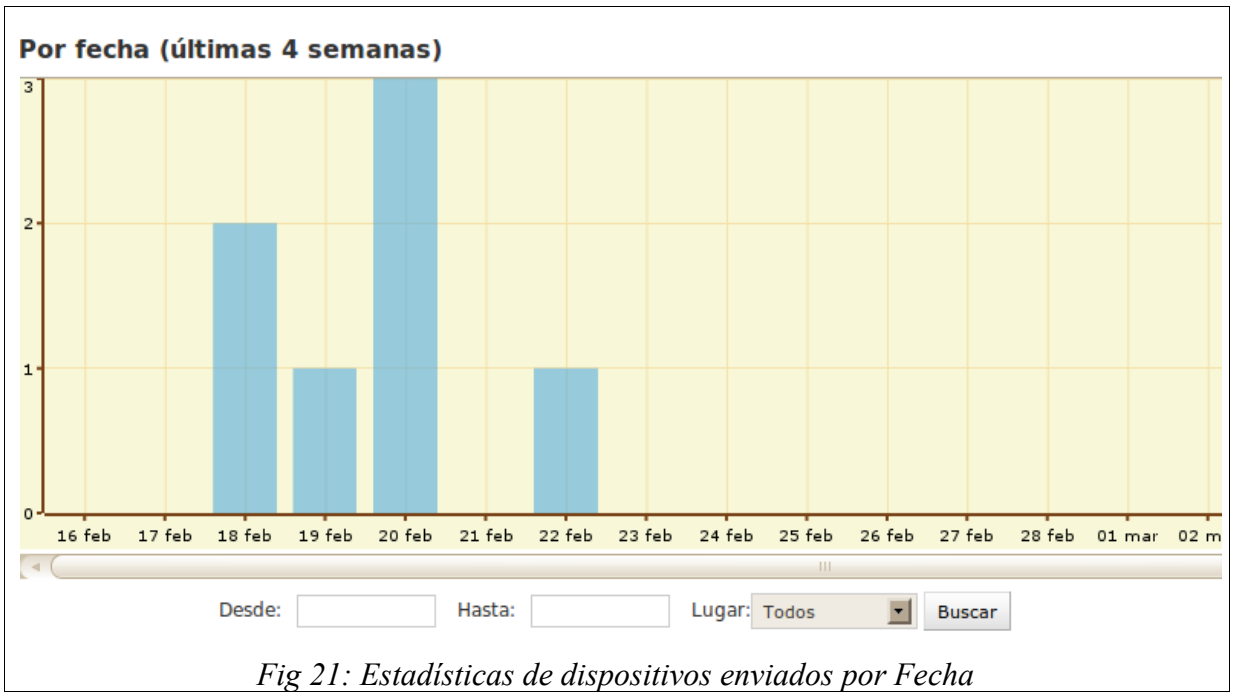


Fig 21: Estadísticas de dispositivos enviados por Fecha

- Por Hora: Por defecto muestra todas las franjas horarias, separadas por hora, de todos los lugares a partir de la puesta en producción del sistema, pero se puede personalizar para que busque entre dos fechas y para un lugar determinado.

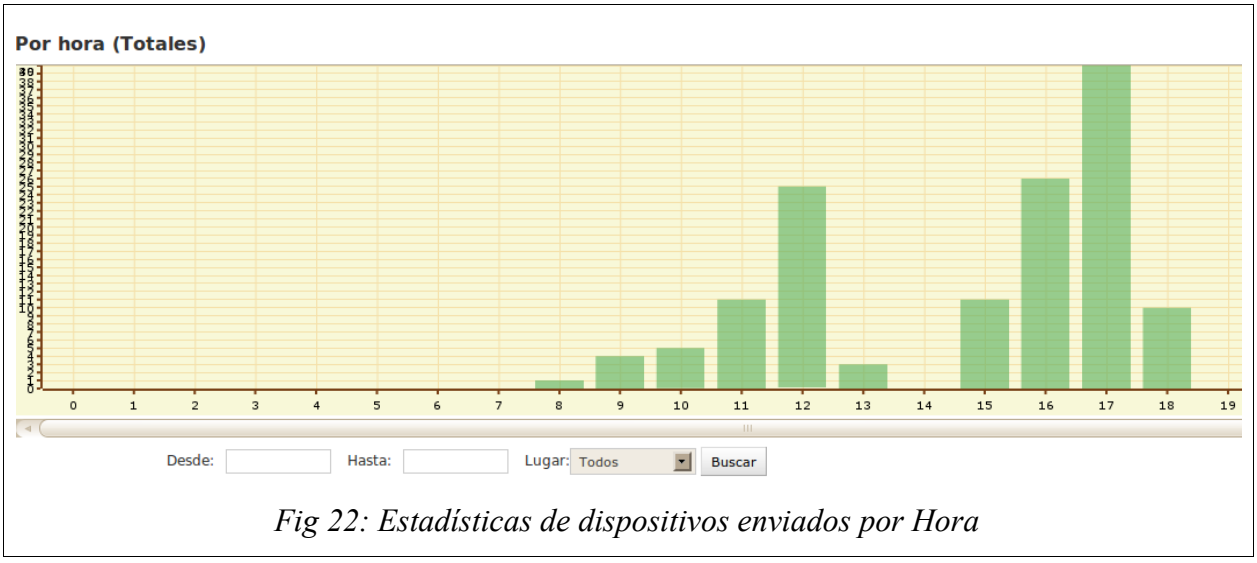


Fig 22: Estadísticas de dispositivos enviados por Hora

- Por Lugar: Muestra un porcentaje de todos los dispositivos encontrados a partir de la puesta en producción del sistema discriminados por lugar.

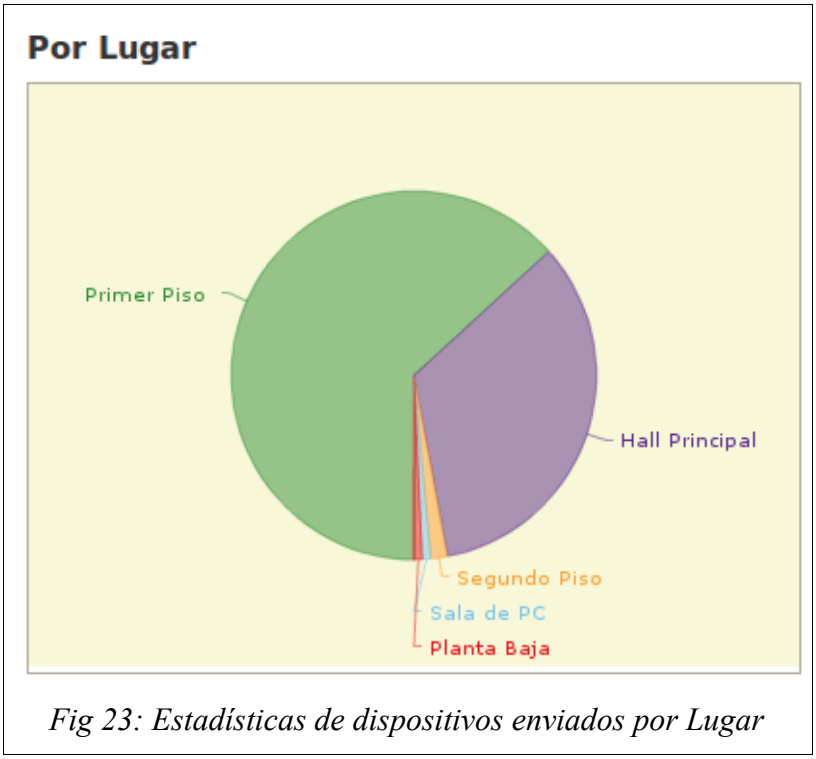


Fig 23: Estadísticas de dispositivos enviados por Lugar

Por Tipo: Muestra un porcentaje de todos los dispositivos encontrados discriminados por tipo a partir de su clases (mayor y menor) desde la puesta en producción del sistema.

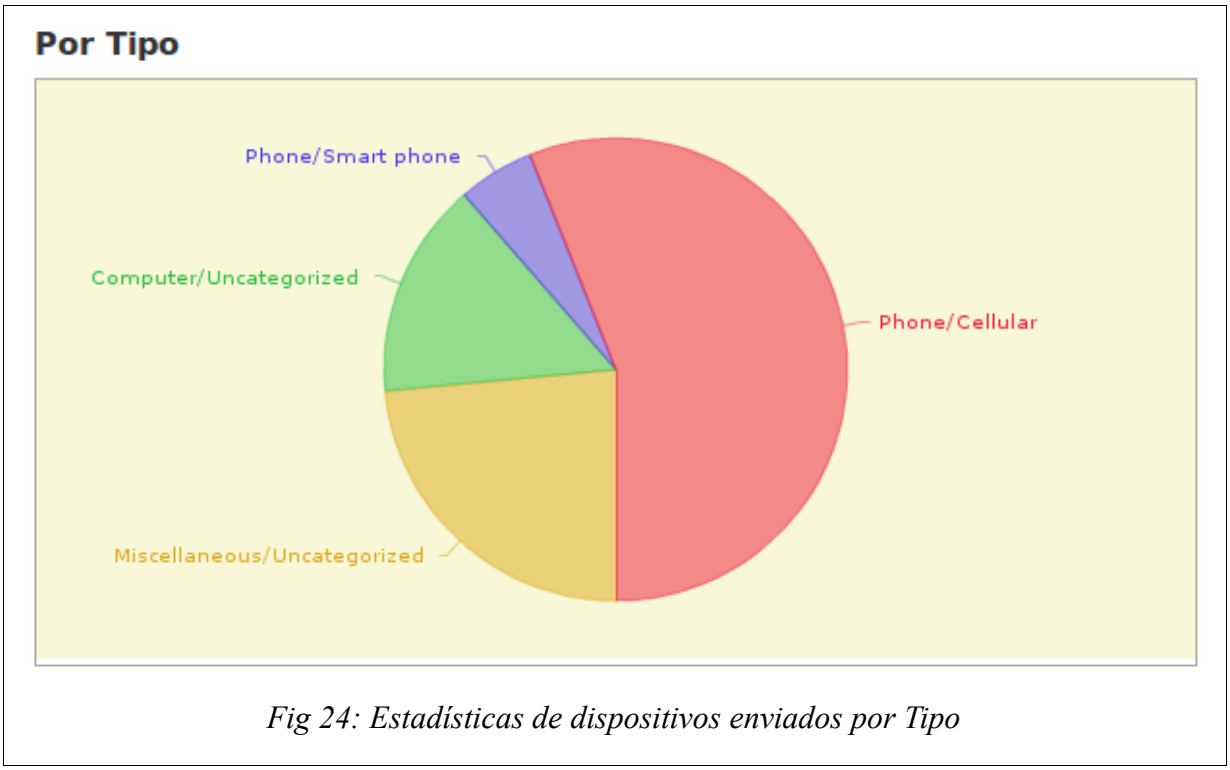


Fig 24: Estadísticas de dispositivos enviados por Tipo

Por otro lado se encuentran los **Datos Enviados**, donde se podrán ver todos los datos que fueron enviados a cada dispositivo junto con su estado (si el mismo fue aceptado o rechazado), su fecha de envío y la ubicación del dispositivo.

El modulo también posee gráficos estadísticos muy similares a los anteriores: por fecha, hora, lugar y tipo, pero este último se refiere a los tipos de datos que se transmitieron, como por ejemplo imagen, texto, contacto, evento, etc.

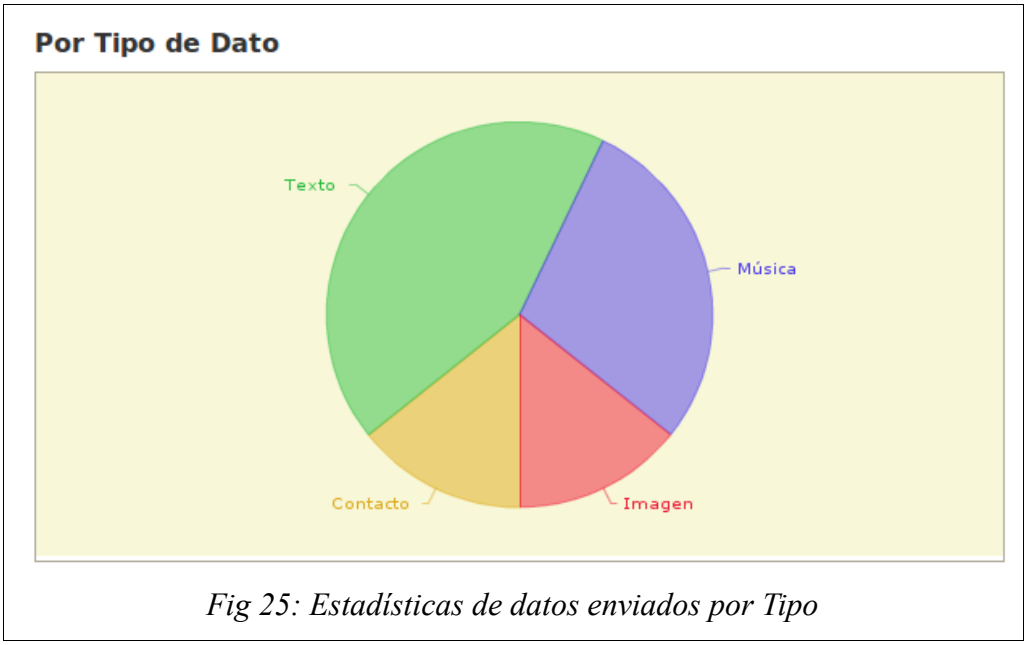


Fig 25: Estadísticas de datos enviados por Tipo

Por último, se tiene información más precisa sobre un **Dato** determinado, detallando por cada uno cuántas veces fue enviado, cuántas aceptado y cuántas rechazado y, si fue enviado alguna vez, se podrán ver gráficos estadísticos por fecha y por hora sobre ese dato en particular.

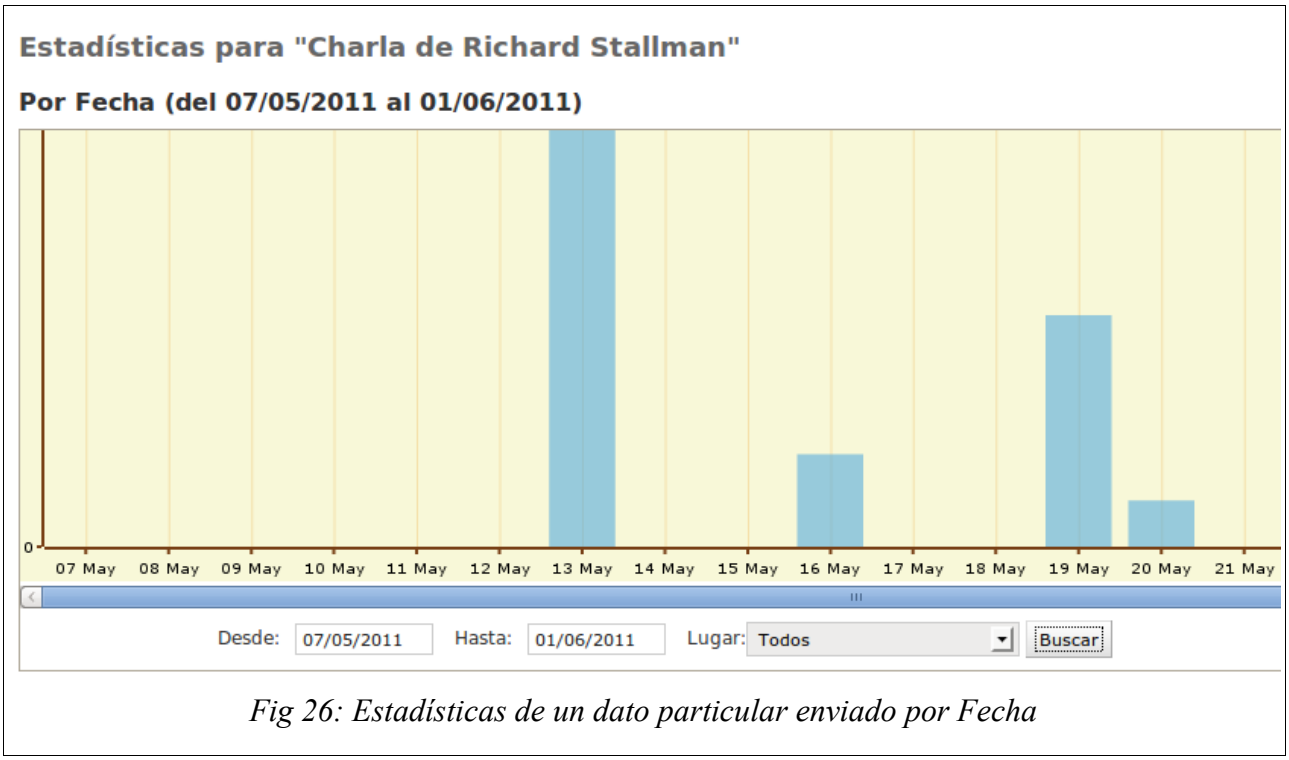


Fig 26: Estadísticas de un dato particular enviado por Fecha

4.6 Decisiones Tomadas

La mayoría de las decisiones sobre el modelo y la arquitectura utilizada se tomaron con el objetivo de incrementar la performance del sistema y para que los modulos principales, Sistema Ubicuo y Administración Remota, sean escalables.

En primera instancia se tomó la decisión de que el sistema operativo huésped del sistema ubicuo sea GNU/Linux por varias razones: una por ser software libre – lo cual posee numerosas implicancias, detalladas en el capítulo 3- , además de la importancia de promover este tipo de software. Pero fundamentalmente fue elegido porque en sistemas Windows la implementación de la pila Bluetooth no permite tener más de un sensor o dongle conectado en la misma máquina huésped. Esto imposibilitaría que se exploten las principales características del sistema: **la concurrencia y el balanceo de carga**. La implementación de dicha pila en GNU/Linux permite tener tantos sensores simultáneos como se desee.

Con la finalidad de hacer un mejor manejo de la concurrencia se agregaron clases intermedias capaces de manejar “hilos” (Threads) que provee el lenguaje.

El modelo de objetos se encuentra estrechamente relacionado con el modelo de datos mediante el ORM elegido, que es SQLObject. Su gran facilidad en el manejo de la persistencia permite que la misma sea totalmente transparente y natural. Además provee un mecanismo de evaluación perezosa (lazy evaluation) para la modificación de objetos ya existentes, que demora la persistencia de los cambios de atributos de un objeto, lo cual puede ser útil en algunos casos ya que no impacta directamente en la performance y el tráfico de red.

Con esta implementación del modelo de objetos con la base de datos se tiene la ventaja de que cualquier cambio realizado desde la administración remota, será inmediatamente detectado por todos los clientes (dongles stations) donde el módulo ubicuo es ejecutado, sin la necesidad de poseer una capa intermedia de abstracción del modelo de datos, ya que lo provee el mismo ORM elegido.

4.7 Performance

Desde el Módulo Ubicuo se trabajó especialmente en la performance global del sistema, ya que éste es el que va a suministrar información a los dispositivos móviles y es el que aporta un gran conocimiento al recinto con importantes datos que serán interpretados por el módulo Web de “Administración Remota”.

Un gran inconveniente que presentan las soluciones con esta tecnología es que cuando un sensor está en proceso de “descubrimiento” de equipos, no se lo puede utilizar para realizar un envío. Esta característica impacta de manera directa en la performance general del sistema. Teniendo en cuenta que los dispositivos detectados están al alcance del sensor por un corto lapso de tiempo, fue imprescindible realizar mejoras en este aspecto. A su vez, por cada dispositivo y cada acción realizada y evento detectado se almacena en la base de datos; es por estas características del sistema que se ha pensado como un sistema escalable y concurrente.

El módulo ubicuo está preparado para operar con una cantidad variable de sensores o dongles, claro está que cuantos más se utilicen, mejor será la performance obtenida en la transmisión y recolección de información de los dispositivos detectados. Mientras un sensor es el encargado de realizar la búsqueda o descubrimiento constante de dispositivos, hay otros encargados de la transmisión de la información hacia los

dispositivos detectados por el anterior. Cada vez que se descubre alguno de manera concurrente, se almacena en la base de datos y se verifica qué información hay que transmitirle (recordar el Cap. 4.5.2.1 en donde se explica el funcionamiento del módulo en detalle). Luego comienza el envío de la información a cada dispositivo, que se logra nuevamente de manera concurrente y se realiza un balanceo de carga sobre el pool (conjunto) de sensores asociados a la ubicación en donde se detectó dicho dispositivo, que se realiza dependiendo de cuántas transmisiones tenga en dicho instante cada uno. Esto es realmente importante, teniendo en cuenta que en ambientes de alto tráfico de dispositivos, evita la sobrecarga de algún sensor, permitiendo lograr una carga uniforme en base a la cantidad de dispositivos detectados en esa zona. Luego de cada envío, sea exitoso o fallido, dicha información también es persistida para futuras consultas mediante el módulo de Administración Remota.

5 Otras aplicaciones y trabajos futuros

En este capítulo se mencionan otras aplicaciones del prototipo presentado en el trabajo, indicando pautas generales para su implantación. Por otro lado se detallan trabajos futuros basados en la investigación realizada y la experiencia obtenida, orientando a una posible aproximación para su desarrollo.

5.1 Otras aplicaciones

La herramienta que se ha desarrollado no fue pensada para un único contexto, sino que se realizó de la manera más genérica posible con la idea de que permita ser adaptada a múltiples aplicaciones. En esta sección se detallan varias implementaciones posibles aplicadas sobre diversos escenarios.

5.1.1 Módulo de mensajes de emergencia (con manejo de prioridades) ante siniestros, desperfectos edilicios y/o falta de servicios básicos

Podría resultar sumamente útil el desarrollo de un módulo para el manejo de envío de mensajes con prioridades y de cambios de modo del sistema general, como ser, ante cualquier altercado de manera manual o automática, el sistema podría operar en modo “emergencia”, por así llamarlo, y enviar los mensajes acordes a dicho altercado o siniestro. De esta manera, se podrían evitar muchos inconvenientes y en el caso que sea necesario hasta provocar un eventual desalojo del recinto.

Podría ser interesante, además, para la realización de simulacros de incendio o cualquier otro siniestro en donde mediante el sistema se le informe a los circulantes que deben abandonar el edificio indicando la manera de hacerlo dependiendo de donde estén ubicados dentro del lugar. El sistema enviaría los mensajes en cada sector donde cuente con sensores (cuantos más sectores cubiertos tenga, más precisa será la evacuación). Para cada sector se podrían indicar las salidas de emergencia o dar indicaciones de cómo actuar.

Una aproximación aún más detallada de la posible expansión del sistema sería que, además del modo emergencia, se establezcan responsables por piso o zona y mediante su dispositivo el sistema le indique los pasos a realizar para evitar que ante la

urgencia y nerviosismo se olvide alguno como responsable de su zona o piso.

Así como hay diferentes modos de debug, se podría implementar el modo de emergencia mediante un mecanismo similar. En la forma manual parece una simple extensión de lo ya desarrollado. Pero lo más interesante de estos sistemas es poder darle la capacidad de ser autónomos y suficientes para detectar estas anomalías en el recinto.

Por dar un ejemplo, el sistema podría estar alimentado por UPS que ante la falla de corriente eléctrica continuaría su funcionamiento normalmente y teniendo en cuenta que los dispositivos móviles cuentan con su propia batería, esa perspectiva estaría totalmente cubierta. Con equipos estándar de UPS bastaría para al menos la emisión de todos los mensajes necesarios, ya que los dongles son alimentados por las mismas estaciones a las cuales están conectadas.

¿Cómo darle autonomía y detección automática?

Como ya mencionamos la autonomía se la damos con alimentación por UPS (si es necesario el continuo funcionamiento se podrían agregar generadores eléctricos) y la detección automática mediante diferentes mecanismos dependiendo de qué es lo que se quiere detectar de esta forma. Para seguir con el ejemplo anterior (corte de luz) la mayoría de los UPS estándar poseen un puerto serial (u otro puerto) para la comunicación con la estación. En dicha comunicación se envían señales constantemente de los eventos que la UPS detecta, como ser, si la corriente es estable, la pérdida y/o el regreso de la misma. Todo esto se envía mediante un cable serial conectado (Interface RS232) a nuestra estación que mediante un servicio del Sistema Operativo detecta y actúa en consecuencia de lo que ocurre. El qué hacer se le indica dependiendo de la necesidad. Un ejemplo de este servicio puede ser el GenPower (Monitor UPS and handle line power failures).[18] De esta manera, se detectan automáticamente los diferentes eventos eléctricos, y en base a ello, poder actuar automáticamente (sin intervención humana); el sistema, al detectar una pérdida de corriente, podría cambiar a modo “emergencia” y actuar en base a dicho evento detectado.

Otra posible automatización de eventos sería mediante la comunicación con sensores detectores de humo o alarmas de incendio y de manera similar a la planteada anteriormente, poder detectar dichos eventos, cambiar de modo y emitir instrucciones de cómo actuar a todas las personas que se encuentran en el recinto dependiendo de su

18 http://packages.qa.debian.org/g/genpower.html

ubicación actual. Hay sensores detectores de humo (o módulos de control) que actúan de manera similar a la comentada por las UPS, enviando señales por puertos seriales, paralelos, USB y hasta Ethernet (RJ45).

5.1.2 Seguimiento personalizado

Muchas organizaciones necesitan realizar un seguimiento permanente del personal, que se puede lograr mediante diversos tipos de mecanismos: tarjetas magnéticas, de aproximación, huellas digitales, etc. Sería una buena alternativa hacer uso del sistema para estos fines, ya que hoy en día la mayoría de las personas poseen celular con servicio de Bluetooth, que son los requisitos mínimos para la implantación del sistema.

Una posible implementación sería asociar cada dirección MAC de los dispositivos con una persona física (de un sistema ya existente o desarrollar un pequeño módulo propio). Lo único que habría que incorporar es la asociación previamente descripta y un reporte y búsquedas de lo almacenado por fechas, horas, etc. (como los ya desarrollados en el módulo de estadísticas). El sistema está casi preparado para esta implementación, y con unos cambios mínimos, se podría llevar a cabo sin ningún problema. En las zonas de detección no se configuraría ningún Sender (solo Discovers encargados de la detección y almacenamiento de lo registrado por zonas) y de esta manera estaría adaptada de manera sencilla para realizar dicha implementación.

5.1.3 Envío de información según el idioma del usuario

En lugares turísticos, los visitantes provienen de diferentes países y hablan diversos idiomas. Sería muy conveniente aplicar una metodología de envío de la información que, según el idioma del usuario del dispositivo móvil, la misma se encuentre en su idioma nativo. Como resultado de esto, por ejemplo, un turista alemán puede visitar un lugar histórico en Argentina; a medida que recorre los distintos salones o cuartos que posee el recinto, se le enviaría información sobre cada uno de ellos directamente en alemán, sin la necesidad de tener un guía o traductor a su lado.

Actualmente, en algunos museos del mundo (como por ejemplo el Palacio de Versalles en Francia), se le entrega al turista un dispositivo electrónico (ya configurado con el idioma del visitante) con un teclado numérico que permite ingresarle el código de

cada salón para que se reproduzca un audio acerca del lugar. Esto requiere un equipamento especial restringido a la cantidad de visitantes.

Una aplicación posible para realizar algo similar con el prototipo sería que, cuando el turista se encuentre en la entrada del lugar, renombre su dispositivo por unos minutos con un código perteneciente a su idioma, como por ejemplo “es”, “en”, “it”, “fr”, etc. La aplicación asociaría la MAC del dispositivo con el idioma, y a partir de ese momento se le enviará la información de cada salón o cuarto que visite en su idioma. Lo beneficioso de esto es que se pueden agregar muchos idiomas para un mismo contenido, lo cual sería difícil de tener impreso en alguna parte del lugar y que sea accesible para todos y no sería necesario tener un dispositivo electrónico para cada turista, ya que utilizaría su propio celular.

5.1.4 Envío de información selectiva por dispositivos

En diferentes implementaciones puede ser muy útil poder discriminar la información que se le envía a los dispositivos, no solo por la ubicación en donde son detectados como hasta el momento, sino también hacerlo dependiendo del dispositivo detectado. Es decir, que se podría discriminar la información enviada a dispositivos individuales o grupos de dispositivos cuyos usuarios tengan alguna característica en común.

Un ejemplo concreto de una posible implementación podría ser en un edificio, en el que a cada persona que ingresa durante fechas determinadas se les envíe el detalle de las expensas (mayormente se paga por m^2 y pueden haber diferentes tamaños de departamentos influyendo en el monto a abonar por cada uno), indicando el monto, vencimientos, etc. Dicha información se podría repetir N veces hasta que se detecte el pago de la misma, por dar una idea concreta. Por otro lado, y siguiendo con un mismo ejemplo, se podría enviar a aquellos dispositivos que pertenezcan solo a dueños (no inquilinos) un evento y/o recordatorios de reuniones de consorcio, etc.

Para poder realizar esto, en el sistema simplemente habría que modificar la forma de asociar el dato a enviar no solo con la ubicación, sino también con un listado de dispositivos o grupos discriminados por su dirección MAC. Para ello, habría que cargar previamente las direcciones de cada dispositivo, asociarlas con un rol o grupo y luego, en la carga de los datos a enviar, se asociarían con N dispositivos o grupos. A la hora del

envío, cuando se detecta un dispositivo, se obtiene la información que tiene asociada el mismo y se le envía o no dependiendo de las diferentes políticas que se asocien (repeticiones, lista blanca o lista negra, etc).

Hasta aquí se explicaron algunas posibles implementaciones del prototipo para diferentes escenarios con un mínimo cambio en el código fuente. A continuación se mencionarán posibles aplicaciones cuya implementación implica modificaciones significativas del prototipo.

5.2 Trabajos futuros

En la presente sección se describen nuevas implementaciones partiendo del prototipo dando una aproximación obtenida a partir de la experiencia con la tecnología y los estudios realizados. Estos trabajos pueden ser tan amplios e innovadores como se pretenda, y cada resultado obtenido puede abrir nuevas ramas de investigación dando origen a nuevos trabajos futuros. A continuación se detallan algunas de las posibilidades más destacadas.

5.2.1 Integración con nuevas tecnologías y diferentes niveles de seguridad

Hay ciertas limitaciones que promueven las nuevas tecnologías de celulares como los SmartPhones, los cuales limitan la información que brindan al medio por cuestiones de seguridad. Los dongles que los detectan cuentan con una mínima información. Muchas veces no se cuenta ni con el nombre y mucho menos con los servicios que este dispositivo es capaz de aceptar o brindar. Esto es realmente necesario para la transmisión ya que si no se sabe si acepta un servicio no se podrá hacer uso del mismo y hasta se ignora por qué puerto es capaz de recibirlo.

Aproximación

Una alternativa del presente trabajo de investigación es agregarle la posibilidad de hacer uso de protocolos y servicios autenticados o emparejamiento automático (colocando en cada extremo los mismos 4 dígitos) para que desde dichos dispositivos se permita la recepción de la información deseada en transmitirle.

En el caso de los Blackberry no solo se debe tener emparejado el dispositivo móvil con la máquina huésped (más precisamente tiene que estar emparejado con el sensor que le envía) de los dongles sino que también desde el móvil se deberá habilitar momentáneamente la publicación del servicio Object Push utilizado por el sistema ubicuo.

El caso de los iPhones es aún mas complejo, ya que es una comunidad muy cerrada y su Sistema Operativo parece muy acotado en cuanto a la coexistencia con otros dispositivos que no sean de la propia partida. Es decir que no aceptan intercambio vía Bluetooth con otros dispositivos, sino solo con otros de su misma especie. Al igual que con las comunicaciones, estos dispositivos tienen un alto grado de rechazo a la instalación de aplicaciones externas. De todas maneras hay técnicas para la instalación de software de tercera partida (third-party), tal como el BlueNova, y de esta manera el sistema ubicuo funciona con total normalidad.

5.2.2 Módulos Interactivos

Una apuesta interesante es acoplarle al prototipo un nuevo submódulo en el módulo ubicuo que permita la recepción de información por parte de los dispositivos próximos a los sensores. De esta manera se podría crear una interacción más que interesante siendo un sistema de emisión y recepción de información, y pudiendo el circundante interactuar de diversas maneras con el sistema.

Aproximación

Así como hay sensores dedicados al descubrimiento de equipos, una aproximación podría ser la de tener dongles dedicados a la recepción de información determinando la interacción del usuario. Esto podría lograrse mediante la instalación de software en el cliente capaz de emitir diferentes mensajes vía Bluetooth de manera tal que sea transparente al usuario y así se puedan codificar las diferentes opciones que el sistema acepte como entrada, dando como resultado un sistema interactivo incrementando su rango de operaciones de manera considerable.

5.2.3 Aplicaciones Clientes para SmartPhones

Debido a la mayor seguridad que traen este tipo de dispositivos, algunos con su configuración y software de fábrica no publican los servicios que necesita el Módulo Ubicuo para poder enviarle información (aunque la mayoría son detectados, y se almacena a fines estadísticos todo lo encontrado). Por consiguiente, es una buena práctica desarrollar aplicaciones para los Sistemas Operativos móviles más usados. De esta manera, la información que se desea enviar podría ser accesible también por este tipo de dispositivos que cada vez son más frecuentes. La finalidad de los clientes instalados en los Smartphones es lograr la comunicación con el Módulo Ubicuo.

Aproximación

Para realizar lo descripto anteriormente, sería necesario desarrollar una aplicación compatible para cada Sistema Operativo móvil más utilizado. Seguramente sea necesario desarrollar más de una versión del sistema ya que no todos los sistemas operativos de este tipo soportan las mismas aplicaciones ni acceden a las componentes de hardware de la misma manera (sensores Bluetooth por ejemplo).

5.2.4 Desarrollo de una capa intermedia entre el modelo de objetos y de datos

En la implementación de la persistencia de los objetos del prototipo se optó por una mayor flexibilidad y un alto grado de dinamismo entre los módulos que lo componen. En algunas situaciones, esta política podría no ser tan favorable. En la implantación del sistema está previsto que se utilice la misma red de datos que posea el establecimiento (en caso de ya poseer una) y por lo tanto se debe adaptar y acoplar a ella. En los casos en los que el establecimiento cuente con equipamiento antiguo u obsoleto y al ser uno de los pilares del prototipo su bajo costo de la puesta a punto, el hecho de que las actualizaciones del sistema en base a la persistencia sea constante y transparente podría ser una contra si tenemos en cuenta que se decidió implantarlo con una arquitectura cliente servidor en donde la máquina huésped de los sensores (dongle station) sea distinta del motor de base de datos, y por consiguiente, se utilice la red como medio de

comunicación.

En redes con dominios de colisiones muy grandes, donde la red esté compuesta por concentradores obsoletos y de baja performance haciendo que grandes redes de datos tengan un alto grado de retransmisiones debido a lo mencionado o al mal diseño de la topología, la performance se podría ver afectada, dependiendo del tráfico y de la estabilidad de la red.

Aproximación

En estos casos particulares se podría implementar una capa intermedia entre el modelo de objetos y el de datos que se encargue de la comunicación entre ellos y no sea el ORM quien lo haga. Es decir, una posible implementación de esto sería separar el modelo de datos del de objetos y construir dicha capa para que el tráfico de red sea el mínimo posible. Si bien el tráfico en bytes generado no es significante, se podría optimizar para estos casos puntuales anunciados en la introducción. De esta manera, dicha capa sería la encargada de la comunicación entre ambos modelos y de dicha persistencia (mediante el ORM). Se estaría minimizando la cantidad de consultas efectuadas y por ende el tráfico de red. Como desventaja de la implementación de dicha capa es que se perdería cierto dinamismo y transparencia de los módulos en su interacción con la base de datos, teniendo que pautar puntualmente el manejo de la persistencia y de las modificaciones de los modelos.

6 Experiencia

Al comienzo de este capítulo se evalúa el impacto de la transmisión vía Bluetooth en la salud de las personas según se ha mencionado en el Cap. 2.10. Luego se detallan todos los pasos realizados para la puesta a punto del prototipo desarrollado junto con los problemas que han ido surgiendo y las soluciones elegidas para el correcto funcionamiento del mismo. Y por último se muestran los resultados estadísticos que se obtuvieron a partir de la implantación del sistema en un cierto período de tiempo.

6.1 Cálculos estimados sobre la implementación

Para la implementación realizada se utilizaron sensores Clase 1 que están dentro de los valores permitidos según los entes de regulación.

Se utilizaron dongles Clase 1 con 100 mW de potencia :

Potencia = 100 mW

Ganancia antena dBi = 1

En primer instancia, se cumplen con las normas internacionales certificadas por la FCC y la CE según fabricante.

Por otro lado, y aún más importante, se tiene el estudio del impacto en la salud de los transeúntes. Como ya se ha mencionado en la Tabla 4, para la frecuencia utilizada por los sensores (2.4Ghz) el límite de exposición recomendado es de 10W/m2 (Densidad de Potencia).

Para realizar el cálculo de la Densidad de Potencia que servirá para determinar si se está por debajo o no del umbral recomendado por la OMS (en particular por la **ICNIRP**), se tendrá en cuenta la potencia y un punto cercano al origen para determinar cuántos W / m2 se están emitiendo: [26]

$D = P / (4 * \Pi * d^2)$

D = Densidad de Potencia

P = Potencia del dispositivo en Vatios

Π = Pi

d = Distancia en metros

Nota: se toma como punto arbitrario a 1 metro del sensor (cuanto más lejos se tome el punto, menor será la densidad de la potencia)

A saber:

Si $1W = 1000mW$ entonces $100mW = 0.1W$

Luego:

$$D = 0.1\ W / (4*pi*d^2)$$

$$= 0.1\ W / (4*3.1415 * 1m^2)$$

$$= 0.1W / 12.566\ m^2$$

$$= 0.007957982\ W/m^2$$

Según lo publicado por la OMS, el valor máximo es de 10 W/m2, por lo cual se puede concluir que no solo se está por debajo de dicho valor, y demostrando que no es nocivo de manera alguna para la salud, sino que se está 1256 veces por debajo de dicho valor.

6.2 Pruebas de campo

La mejor forma de mostrar el funcionamiento del prototipo, más allá de las pruebas realizadas en base a distintos dispositivos obtenidos, mediciones de velocidad y performance, es realizando una puesta en producción del prototipo en entornos reales, con personas que atraviesan la red de sensado, que pueden o no saber de la existencia del mismo, y observar los comportamientos generales y particulares del mismo.

En esta sección se muestran y analizan los datos relevados durante el 13/05/2011

hasta el 29/06/2011 en el Hall Principal del Edificio Tres Facultades donde actualmente hay oficinas de las Facultades de Ciencias Jurídicas y Sociales, Psicología y Humanidades de la UNLP, en una suerte de prueba piloto, y entre el 30/06/2011 y el 16/9/2011 en el Subsuelo del Edificio Reforma Universitaria (ex Jockey), donde actualmente funciona la Facultad de Ciencias Jurídicas y Sociales. Ambos edificios tienen acceso en la calle 48 entre 6 y 7 de la Ciudad de La Plata, Buenos Aires, Argentina. También se colocó otra estación huésped (dongle station), desde el 29/08/2011 en la entrada del Edificio Reforma Universitaria sobre calle 48 (solo como descubridor, sin envío) para detectar el volumen de tráfico del principal ingreso a dicho edificio.

En esta prueba de campo se priorizaron cuestiones de seguridad ante todo. La información que se envía y las estadísticas relevadas en su mayoría son con fines de seguridad edilicia.

6.2.1 ¿Por qué la elección de cada lugar?

La elección de cada zona en la que se desplegaron los sensores tiene una justificación para el proyecto:

- Hall Principal: Este fue el primer lugar en donde se implementó una primer aproximación del prototipo, fue por una cuestión de accesibilidad del sistema, tanto a los sensores como a la máquina huésped de los sensores (dongle-station). En este lugar se hicieron los primeros ajustes del sistema.
- Subsuelo (Ex Jockey): Este lugar fue estratégicamente buscado por la variedad de espacios y servicios con los que cuenta esta planta y la alta circulación en consecuencia de ello. Esta zona cuenta con:
    - Dos posibles accesos (por calle 48 y por calle 49)
    - Sala de profesores
    - Cajeros interactivos para la marca de asistencia del personal
    - Más de 10 aulas
    - Buffet
    - Dos ascensores
    - Escaleras distribuidas por dos zonas distintas

- Baños

- Ingreso por calle 48: La idea de colocar sensores en este ingreso es para obtener una estadística del flujo de personas que ingresan y egresan por este lugar, que es el de mayor circulación de la Facultad.

6.2.2 Puesta a punto

6.2.2.1 Configuración inicial

El sistema fue configurado de tal forma que a los dispositivos recurrentes los almacene cada 10 minutos y así evitar una gran cantidad de tuplas insertadas para aquellos que permanecen mucho tiempo bajo el alcance de los sensores. Además fueron configuradas algunas variables, como por ejemplo el nombre de los sensores con las siglas de la Facultad “FCs.JyS – UNLP”. De esta manera, un dispositivo que reciba datos de los dongles del sistema, visualizará dicho nombre antes de aceptarlo.

6.2.2.2 Política de Broadcast

La política elegida es la de Lista Negra. Como se ha mencionado en el Cap. 4.5.3.3, la elección de esta política es simplemente cambiando la configuración mediante el sistema de administración remota (Web). Al ser la puesta en producción de un prototipo, se considera que se obtendrán más y mejores resultados en cuanto a la recolección de información si se plantea esta política por sobre la de Lista Blanca, en la cual los usuarios se tienen que subscribir explícitamente mediante el nombre de su dispositivo (Ver Capítulo 4.5.2.2)

6.2.2.3 Datos a enviar

Se han configurado tres datos que se consideran importantes para el envío de la información. Como se ha mencionado en el Cap. 4.2, la idea de esta prueba del prototipo es la recolección de datos con fines estadísticos y el envío de información con diversos formatos para personas con capacidades diferentes. Los formatos elegidos fueron principalmente imagen y audio; también se envió el contacto con la información de la Facultad. Los datos enviados fueron:

- **Contacto de la Facultad:** Se envió en formato estándar (VCARD 2.1) los teléfonos, dirección (localidad, código postal, provincia y país), página web y mail institucional. El archivo en texto plano fue el siguiente:

```
BEGIN:VCARD
VERSION:2.1
N:Fac. de Cs. Juridicas y Soc.;;;
FN:Fac. de Cs. Juridicas y Soc.
ORG:;
NOTE;ENCODING=QUOTED-PRINTABLE:=0D=0A
TEL;WORK;VOICE:02214236701
ADR;WORK:;;48 e/6 y 7;La Plata;Buenos Aires;1900;Argentina
LABEL;WORK;ENCODING=QUOTED-PRINTABLE:48 e/6 y 7=0D=0A1900  La Plata
Buenos Aires=0D=0A
ADR;HOME:;;;;;;
ADR;POSTAL:;;;;;;
URL;WORK:www.jursoc.unlp.edu.ar
EMAIL;PREF;INTERNET:info@jursoc.unlp.edu.ar
END:VCARD
```

Cabe aclarar que la VCARD es generada automáticamente por la plantilla creada en la administración remota; el usuario sólo llena los campos correspondientes (Ver 4.5.3.4.1)

Visto por algún intérprete de VCARD se mostraría así:

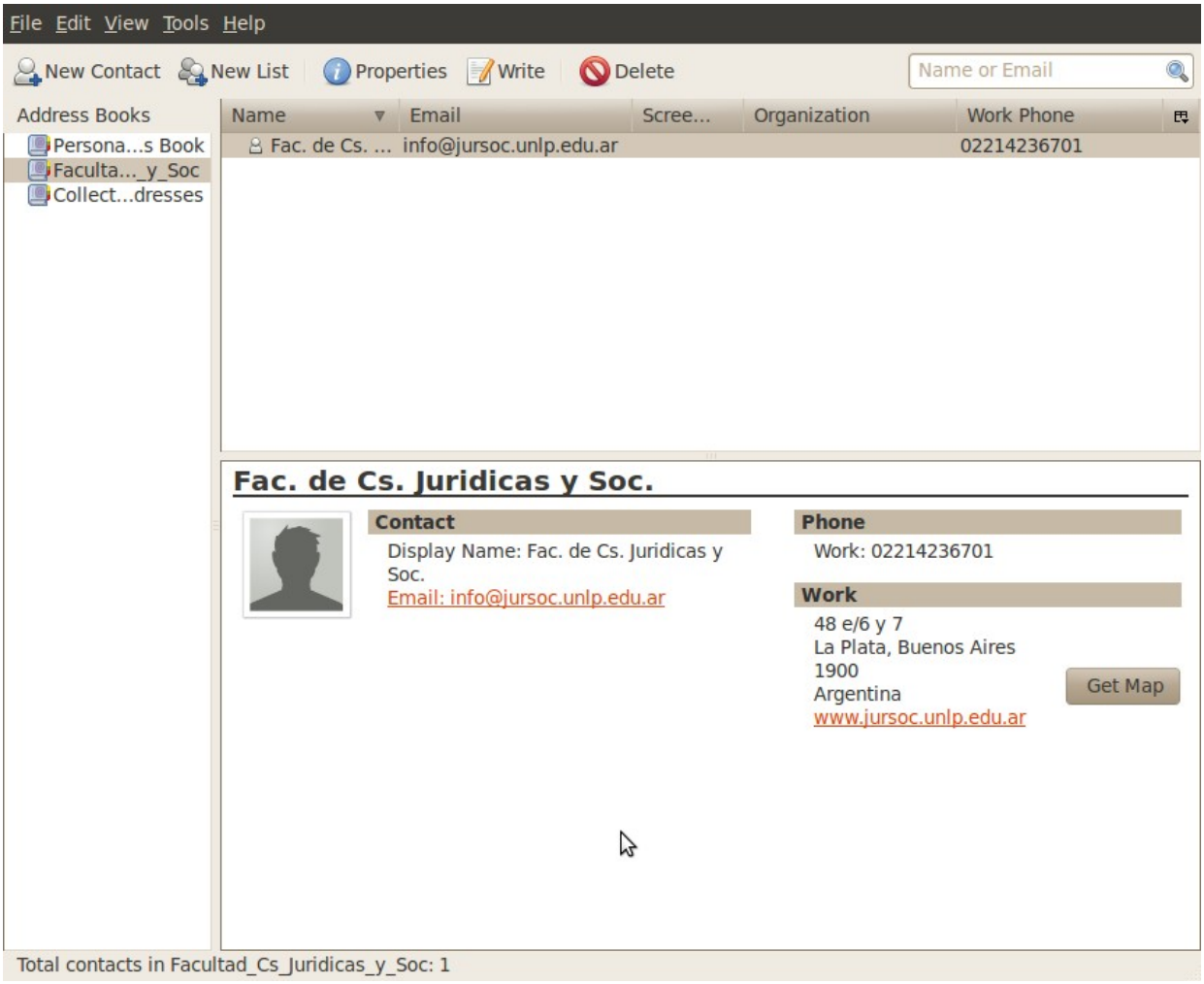


Fig 27: Vcard en Thunderbird

- **Mapa del subsuelo:** Se envió una imagen con el mapa del subsuelo indicando todo lo que se encuentra a su alrededor, haciendo hincapié en las escaleras, ambas salidas (por calle 48 y por calle 49), ascensores, baños y la rampa de discapacitados con la que cuenta el ingreso por calle 49.

Fig 28: Plano subsuelo

Audio Subsuelo: Se envió un archivo MP3 con la grabación de un locutor con la siguiente información:

> "Bienvenido a la Facultad de Ciencias Jurídicas y Sociales, usted se encuentra en el subsuelo del Ex-Jockey. Este nivel cuenta con dos salidas, una por calle 48 y otra por calle 49 que cuenta con rampa para discapacitados. También dispone de dos ascensores, una escalera, luces de emergencia y matafuegos ubicados en el hall principal. Les recordamos que se encuentra prohibido fumar en cualquier ámbito de la Facultad"

6.2.3 Problemas encontrados

Básicamente se han encontrado los dos tipos de errores posibles: software y hardware.

6.2.3.1 Problemas de software

Se han tenido algunos inconvenientes adicionales a los esperados a la hora de programar aplicaciones que interactúan con distintos dispositivos de hardware. A medida que se realizaron las pruebas en un ambiente controlado y con testeos de no más de una hora consecutiva, no surgieron mayores inconvenientes. Una vez que se lanzó el sistema

en las pruebas piloto se detectaron diferentes anomalías que se fueron solucionando. Básicamente, se trató de algún recurso no liberado que en ejecuciones de corto plazo no eran detectados, pero cuando el sistema funcionaba indefinidamente como un servicio continuo fueron apareciendo.

Por otro lado, las librerías utilizadas muestran cierta inestabilidad. Uno de los problemas más comunes es que se detectaban dispositivos, pero cuando a cada uno se le consultaban los servicios que poseían, no retornaba ninguno. Tal vez esto sea un problema compartido entre la librería y el firmware del dongle.

6.2.3.2 Problemas de hardware

Estos problemas han sido los más difíciles de solucionar, por lo que se han clasificado en los siguientes:

6.2.3.2.1 Problemas de hardware defectuoso

Dado que una de las ideas principales del prototipo fue utilizar PCs que estén en desuso por la desactualización tecnológica, es que se puede encontrar hardware defectuoso a corto o mediano plazo. No se ha tenido inconvenientes en las primeras horas de la puesta en marcha pero sí, con el funcionamiento continuo, han aparecido errores de acceso a disco, problemas de memoria o temperatura. Transcurre tanto tiempo desde que se dejó de utilizar una máquina determinada que se pierde el real motivo por el cual se dejó de utilizar y es ahí donde aparecen los diferentes errores.

6.2.3.2.2 Problemas de sensores

Un error importante se encontró en los sensores utilizados para la detección y envío de la información. Por un lado, con algunos sensores adquiridos se ha detectado una problemática inusual: entre ellos poseían la misma dirección MAC, lo cual hizo imposible que funcionen en un mismo dongle-station ya que justamente la creación de los sockets se realiza por dirección MAC y en el sistema cada uno es identificado del resto por dicha dirección. Sin embargo, sí podrían coexistir en diferentes stations, ya que cada una maneja a su grupo de dongles en los diferentes sectores en donde se han desplegado. Se han detectado errores de firmware importantes en el uso intensivo de los sensores. Evidentemente no son dispositivos diseñados para un funcionamiento continuo y tan exhaustivo. Cada ciertos períodos variables de tiempo y dependiendo del tráfico

detectado, el firmware del sensor descubridor colapsa.

Al detectar estas anomalías, en algunos casos, es posible restablecerlos mediante un reseteo de los puertos USB involucrados mediante un programa o script escrito en el "Lenguaje C" invocando a la system call[19] *ioctl*, que se detalla a continuación:

```
/* filename = usbreset.c */
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <linux/usbdevice_fs.h>
int main(int argc, char **argv)
{
	const char *filename;
	int fd;
	int rc;
	if (argc != 2) {
		fprintf(stderr, "Usage: usbreset device-filename\n");
		return 1;
	}
	filename = argv[1];
	fd = open(filename, O_WRONLY);
	if (fd < 0) {
		perror("Error opening output file");
		return 1;
	}
	printf("Resetting USB device %s\n", filename);
	rc = ioctl(fd, USBDEVFS_RESET, 0);
	if (rc < 0) {
		perror("Error in ioctl");
		return 1;
	}
	printf("Reset successful\n");
	close(fd);
	return 0;
}
```

A dicho programa se lo invoca tantas veces como sensores se tengan para lograr una reinicialización completa pasando como parámetro el sensor a reiniciar.

19 Mecanismo utilizado por una aplicación para solicitar una instrucción privilegiada del sistema operativo.

En ciertas ocasiones, realizar esta operación no resulta exitosa, es decir que si un dongle USB dejó de responder, a nivel de sistema operativo no hay mucho más por hacer. El dongle no procesa ningún tipo de requerimiento hasta que no sea apagado y encendido nuevamente. Para lidiar con este problema, se han encontrado varias soluciones posibles:

1. Realizar, de ser posible, una actualización del firmware de los sensores adquiridos que solucionen estos problemas.
2. Adquirir sensores que estén diseñados para soportar un continuo y arduo funcionamiento.
3. Incorporar algún *USB isolation* (aislador USB) que mediante una señal enviada por un puerto paralelo o serial, simule el corte de corriente del puerto USB y luego vuelva a restablecerlo. Son dispositivos de una complejidad electrónica mínima.
4. Al detectar dicha anomalía, realizar un apagado total del dongle-station y mediante un mecanismo automático volver a encenderlo simulando un corte de energía a los dongles USB.

En el contexto de este trabajo, se ha optado por la última opción por varios motivos, pero sobre todo porque no se pretende que sean costos elevados de puesta en marcha. Se desea demostrar que es posible una implementación reutilizando equipamiento y sensores de bajo costo.

A fin de lograr una mayor autonomía, que se ha realizado la siguiente metodología automática:

I. Cada X minutos/horas se ejecuta un cron job[20] que dispara la serie de controles del sistema; si se detecta que algún sensor ya no responde por un eventual colapso del firmware y es inevitable una pérdida de corriente para que el dongle vuelva a un estado de correcto funcionamiento, el dongle-station se apaga.

II. A continuación, mediante un WakeOnLan[21], otra máquina despierta a la que se acaba de apagar; con esto se simula un pequeño corte de

20 Tarea o proceso que se ejecuta en un intervalo de tiempo determinado.

21 Estándar de redes de computadoras que permite encender remotamente máquinas apagadas.

energía remoto y automático en el grupo de sensores, logrando de esta manera una estabilidad continua a un costo menor (sólo unos segundos sin sistema).

III. Al inicio del Sistema Operativo, se ejecuta el sistema automáticamente.

Detalle de la metodología recientemente descripta:

I. En el archivo /etc/crontab se agregó la siguiente línea:

```
*/4 * * * * root  /path/to/blue4as/cronjob.sh
```

Indica que cada 4 horas va a ejecutar el script cronjob.sh, que sería el siguiente:

```
#! /bin/sh
SENSOR_NUMBER=3
cd /path/to/blue4as/
./blue4as stop
sleep 3
id=0
for i in `ls /dev/bus/usb/001/ -r`
do
	if [ $i -gt $id ]; then
		id=$i
	fi
	./usbreset /dev/bus/usb/001/$i
	sleep 3
done
sleep 3
if [ `hcitool dev | grep hci | wc -l` -lt $SENSOR_NUMBER  ]; then
	halt
else
	screen -A -m -d -S "blue4as-"$id nice --20 ./blue4as start
fi
```

Básicamente, lo que realiza este script es detener el servicio de la aplicación Blue4AS, luego reiniciar cada uno de los sensores, después controlar si algún sensor se encuentra inoperante, y si es así realiza el apagado total (halt) y sino ejecuta el sistema en un screen[22] con una alta prioridad.

22 Programa multiplexor de terminales que permite a los usuarios acceder a múltiples sesiones separadas dentro de una sola ventana de terminal o en una sesión de terminal remota.

II. Desde otra máquina sencillamente se ejecuta de forma automática (mediante un cron) el comando “wakeonlan MAC”, especificando la dirección MAC de la placa Ethernet del dongle-station.

III. Al iniciar nuevamente el dongle-station por el paso II, se ejecuta el siguiente script al inicio del Sistema Operativo:

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          blue4as application instance
# Required-Start:    $all
# Required-Stop:     $all
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: starts instance of blue4as app
# Description:       starts instance of blue4as app using start-stop-daemon
### END INIT INFO
################ Parameters  ###################
# path to app
APP_PATH=/home/blue4as/bluetooth
# script name
NAME=blue4as
# app name
DESC=blue4as
# run as
RUN_AS=root
PID_FILE=/var/run/blue4as.pid
################ END ############################
test -x $DAEMON || exit 0
set -e
case "$1" in
  start)
        echo -n "Starting $DESC: "
        screen -A -m -d -S "blue4as-"$id nice --20 $APP_PATH/blue4as start
        echo "$NAME."
        ;;
  stop)
        echo -n "Stopping $DESC: "
        $APP_PATH/blue4as stop
        echo "$NAME."
        ;;
  restart|force-reload)
        echo -n "Restarting $DESC: "
        $APP_PATH/blue4as stop
```

```
        sleep 3
        screen -A -m -d -S "blue4as-"$id nice --20 $APP_PATH/blue4as start
        echo "$NAME."
        ;;
  *)
        N=/etc/init.d/$NAME
        echo "Usage: $N {start|stop|restart|force-reload}" >&2
        exit 1
        ;;
esac
exit 0
```

Con todo lo detallado previamente, se ha logrado solucionar el mayor problema surgido en el proyecto relacionado a la estabilidad de las máquinas que ejecutan el sistema Blue4AS.

6.3 Datos relevados

En base a todos los datos recolectados a lo largo de las pruebas realizadas durante el 13/05/2011 hasta el 16/09/2011, se pudieron sacar las siguientes estadísticas:

6.3.1 Dispositivos únicos y recurrentes

Se realizó el cálculo de los dispositivos que aparecieron una única vez (nuevos) y aquellos que han vuelto a aparecer más de una vez (recurrentes), totalizando lo siguiente:

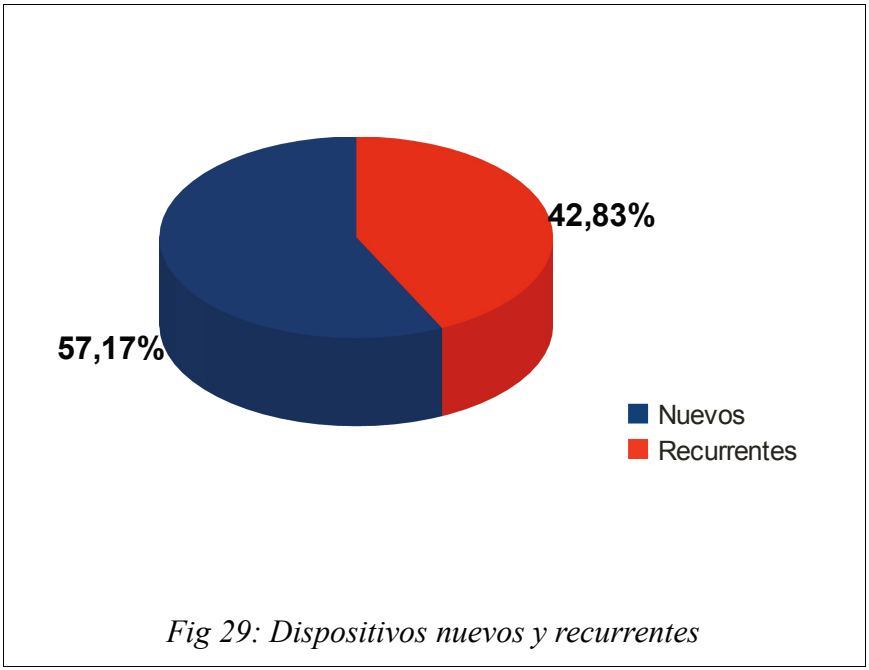


Fig 29: Dispositivos nuevos y recurrentes

Es decir que la mayoría de los dispositivos encontrados estuvieron de paso y no han vuelto a aparecer por los lugares donde se han colocado los sensores, pero de todas formas hubo una gran cantidad de dispositivos recurrentes.

6.3.2 Promedio de dispositivos por día

Se calculó el promedio de dispositivos detectados por día (sin contar los fines de semana) a partir de la tercera semana de Mayo hasta la segunda semana de Septiembre. Además se compararon entre detecciones únicas y totales.

La siguiente tabla detalla los datos de cada semana y en qué lugares fueron detectados los dispositivos según las pruebas realizadas:

Mes - Semana	*Promedio detecciones únicas*	*Promedio detecciones totales*	*Lugar/es*
Mayo - 3	37,2	79,6	Hall Principal
Mayo - 4	51,25	158	
Mayo - 5	24,5	118	
Junio - 1	10,66	44	
Junio – 2	21	85	
Junio – 3	16	168	
Junio – 4	11,66	87,66	
Junio - 5	34,33	111	
Julio – 1	23	59	Subsuelo (Ex Jockey)
Julio – 2	34,2	109	
Julio – 3	32,6	77	
Julio – 4	0	0	
Julio – 5	0	0	
Agosto – 1	28	62	
Agosto – 2	29,6	75	
Agosto – 3	55,2	159	
Agosto – 4	62	256,75	
Agosto – 5	132,66	365,66	Subsuelo (Ex Jockey), Ingreso por calle 48
Septiembre – 1	116	350	
Septiembre – 2	90,6	324,8	
TOTAL	**40,52**	**134,47**	

Tabla 9: Promedio de detecciones únicas y totales por semana.

Con detecciones únicas se refiere a dispositivos encontrados sin contar sus repeticiones durante el día, y con detecciones totales se refiere a dispositivos encontrados contando si se almacenó más de una vez en el mismo día.

El gráfico asociado es el siguiente:

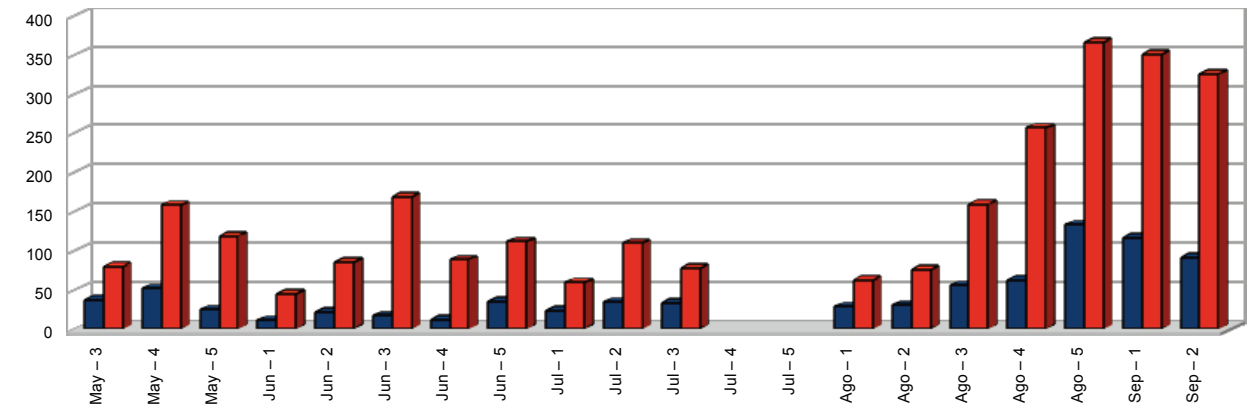


Fig 30: Promedio de dispositivos por día

Se puede observar un gran incremento con el agregado de los sensores en el ingreso por calle 48 y a partir de la tercer semana de Agosto debido al inicio de las cursadas de los alumnos. En las últimas semanas de Julio no se registraron datos debido al receso invernal de los estudiantes, por lo cual la Facultad permaneció cerrada.

6.3.3 Cantidad de dispositivos por tiempo de permanencia

Se contabilizaron los dispositivos por su permanencia en el recinto, calculando sus tiempos de detección para luego incluirlos en subdivisiones de rangos de horas y minutos, como ser: mayor a 1 hora, entre 40 y 60 minutos, entre 20 y 40 minutos, entre 10 y 20 minutos, y menor a 10 minutos.

Para dichos cálculos, se dividió por las dos zonas principales relevadas: el Subsuelo y la Entrada de calle 48. Los resultados para ambos fueron los siguientes:

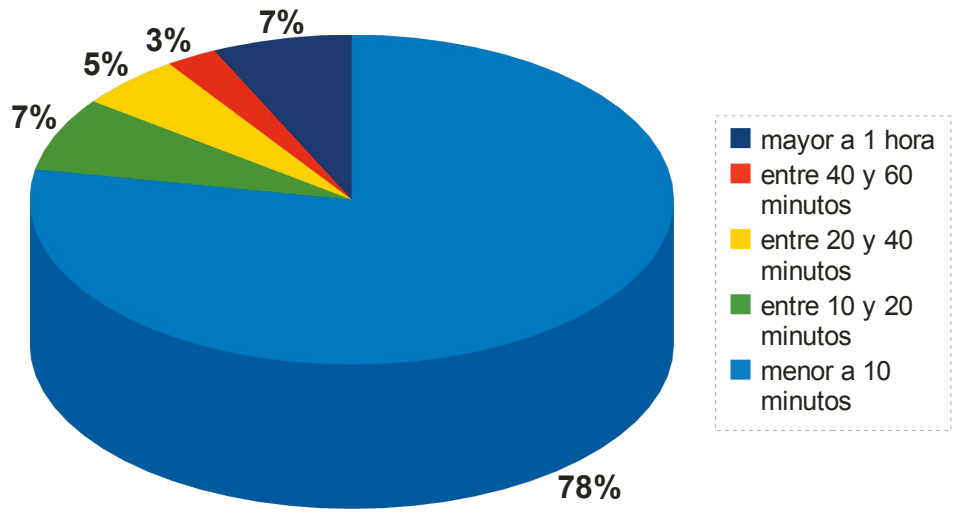


Fig 31: Cantidad de dispositivos por tiempo de permanencia (Subsuelo)

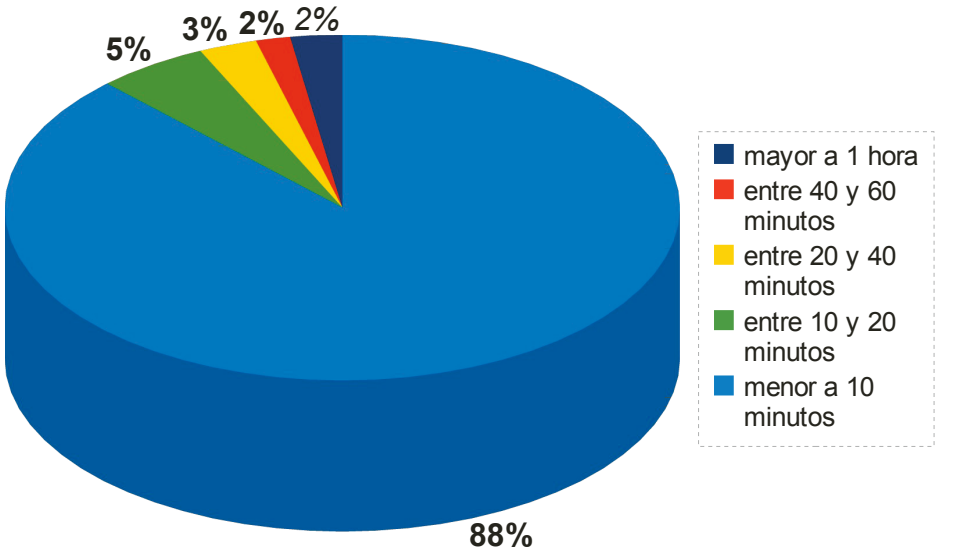


Fig 32: Cantidad de dispositivos por tiempo de permanencia (Entrada calle 48)

Puede verse que la gran mayoría permaneció menos de 10 minutos en el lugar, siendo los mismos lugares de paso, pero también existe un gran porcentaje que permaneció más de una hora, por lo cual los envíos realizados por el sistema pueden

efectuarse con mayor éxito.

Si bien en ambos lugares se han dado casi los mismos resultados, se puede notar que en el ingreso de calle 48 hubo una mayor cantidad de dispositivos que permanecieron menos de 10 minutos, ya que es una zona más de tránsito, y en el subsuelo hubo un mayor porcentaje de dispositivos que permanecieron más de una hora al ser, por el contrario, una zona de mayor permanencia.

6.3.4 Cantidad de envíos totales

Se contó la cantidad de envíos totales con y sin errores, dando los siguientes resultados:

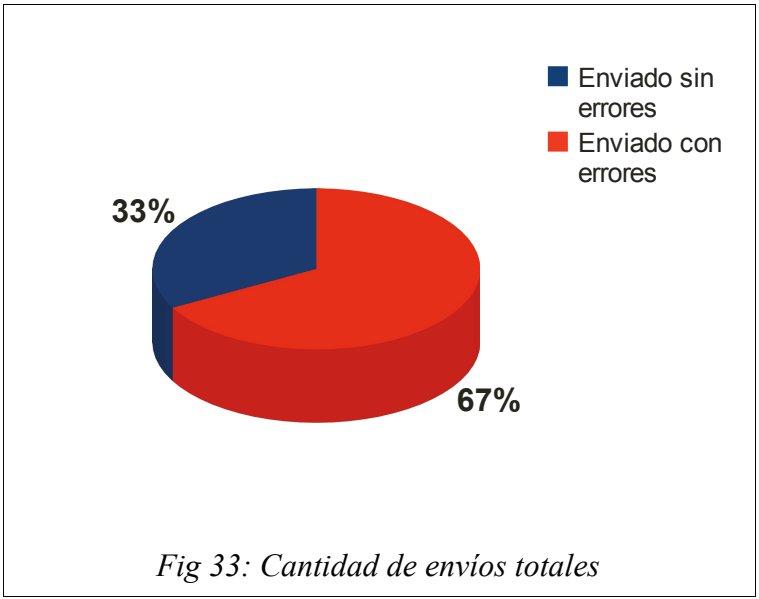


Fig 33: Cantidad de envíos totales

Dentro de aquellos envíos sin errores, se calculó la proporción de datos aceptados y rechazados, reflejados en el siguiente gráfico:

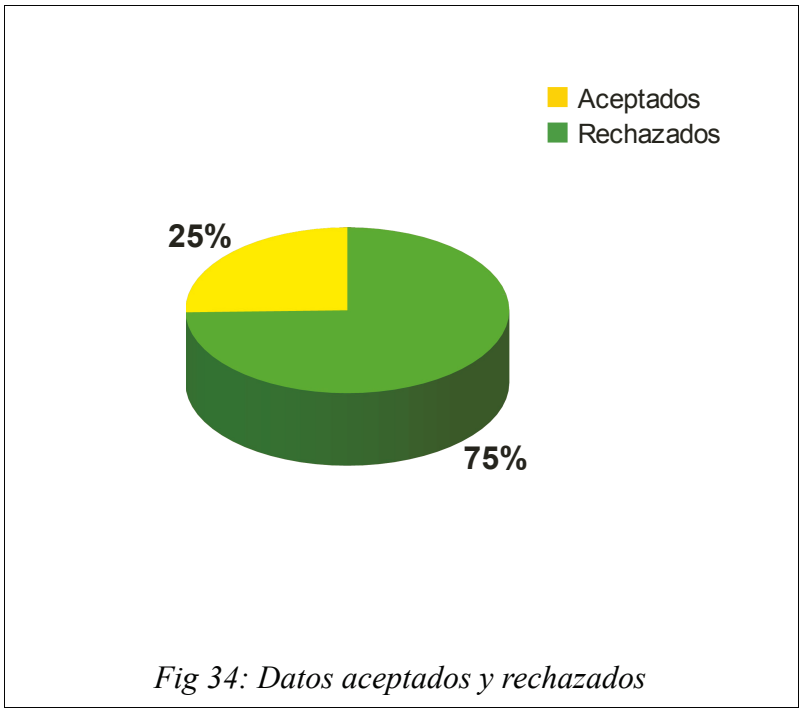


Fig 34: Datos aceptados y rechazados

Y dentro de aquellos envíos inconclusos, se analizaron las diferentes causantes discriminadas por tipo de error, obteniendo el siguiente gráfico:

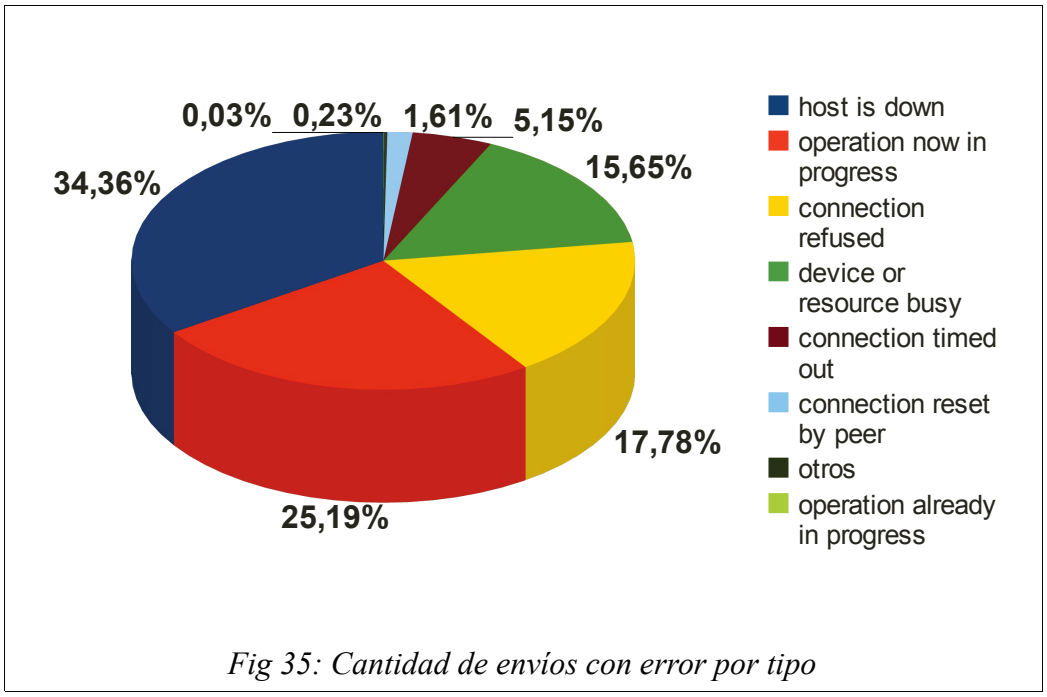


Fig 35: Cantidad de envíos con error por tipo

A continuación se describen los errores detectados:

Tipo de Error	*Descripción*
Host is down	Debido a la circulación de los dispositivos se puede salir del rango y más allá del alcance de la señal
Operation now in progress	Cuando no se toma ninguna acción por parte del usuario (aceptar o rechazar) y el sistema, en un nuevo ciclo, intenta enviar nuevamente la información
Connection refused	En algunos sistemas que requieren emparejamiento para el protocolo usado, se rechaza la conexión
Device o resource busy	El dispositivo se encuentra temporalmente ocupado
Connection time out	Luego de un tiempo de establecido el socket, se cancela la conexión por tiempo de espera agotado
Connection reset by peer	El dispositivo indica que necesita restablecer la conexión
Operation already in progress	El requerimiento de conexión ya se encuentra en progreso para el socket especificado

Tabla 10: Tipos de errores de envío detectados

6.3.5 Datos enviados por lugar

Como los sensores que emiten datos fueron implementados sólo en el Subsuelo y en el Hall Principal, las estadísticas muestran que la mayoría de datos enviados se realizaron en el Subsuelo (92,37%), sumado a que el período de emisión es también mayor. El gráfico asociado es el siguiente:

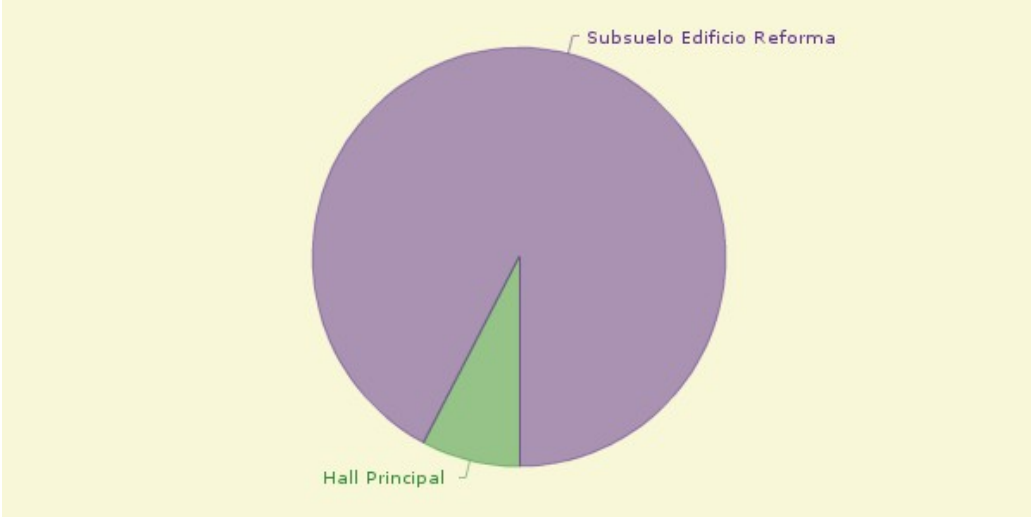


Fig 36: Datos enviados por lugar

6.3.6 Datos enviados por tipo de dato

Los datos a enviar se crearon de distintos tipos para las pruebas, y durante la implementación del proyecto se enviaron según el siguiente gráfico:

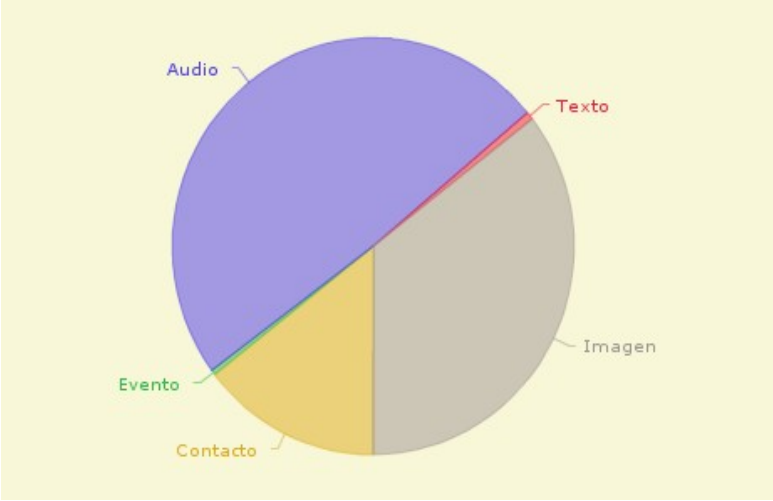


Fig 37: Datos enviados por tipo de dato

El dato de mayor envío fue el de audio (49,09%), seguido de la imagen (35,47%) y el contacto de la Facultad (14,45%). Durante el primer período, también se realizaron pruebas con texto plano (0,58%) y eventos (0,4%).

6.3.7 Dispositivos detectados por hora

El sistema permite contabilizar la cantidad de dispositivos discriminados por hora en todos los lugares. Luego de todas las pruebas realizadas, se dieron los siguientes resultados:

Hora	*Cantidad de dispositivos*	*Hora*	*Cantidad de dispositivos*
0	59	12	960
1	19	13	925
2	68	14	839
3	92	15	893
4	46	16	926
5	148	17	1080
6	74	18	911
7	168	19	716
8	528	20	476
9	768	21	288
10	951	22	116
11	1064	23	52

Tabla 11: Cantidad de dispositivos totales detectados por hora.

Y el gráfico asociado es el siguiente:

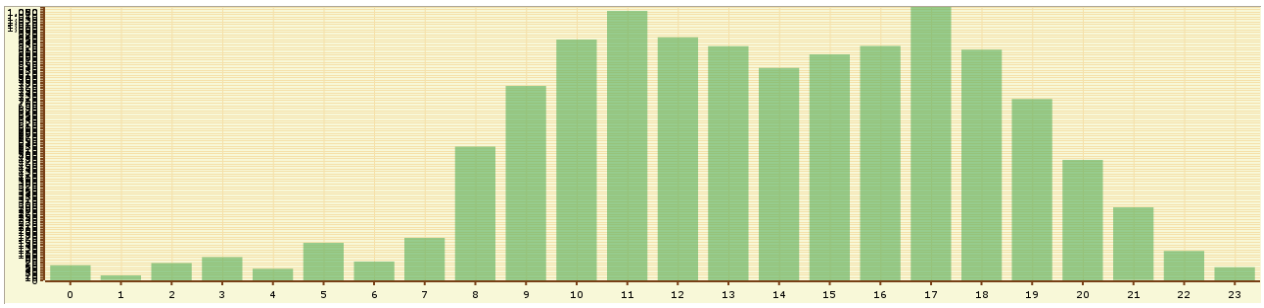


Fig 38: Dispositivos detectados por hora

Podemos ver que los horarios en que más dispositivos han sido detectados son las 11 de la mañana y las 5 de la tarde, pero manteniendo un valor similar desde las 9 de la mañana hasta las 19 hs.

6.3.8 Dispositivos detectados por fecha

Se dividieron los datos de todo el mes de Agosto y la primer quincena de Septiembre, dando como resultado los siguientes gráficos estadísticos:

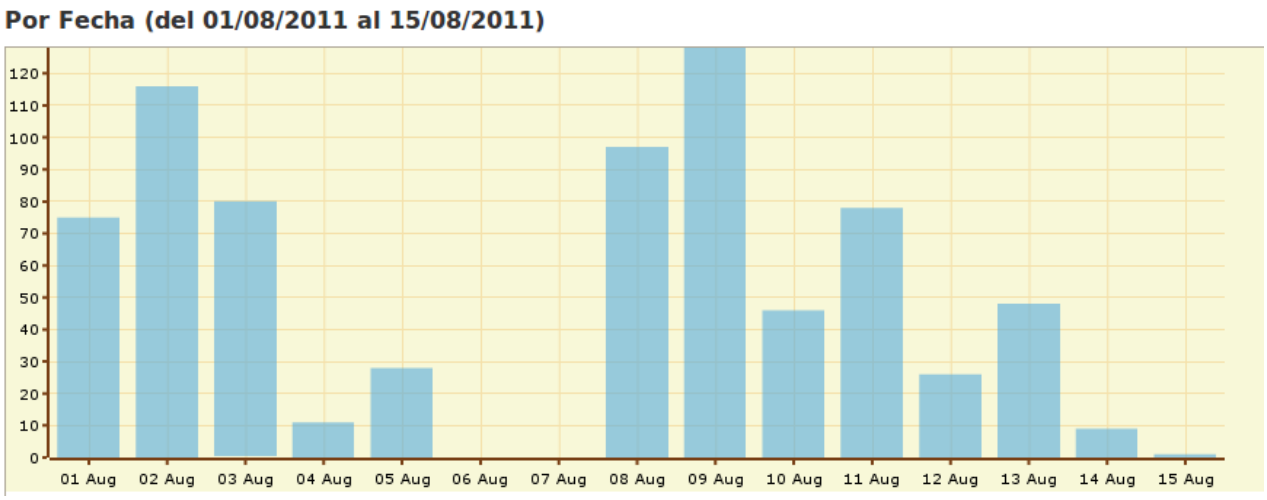


Fig 39: Dispositivos encontrados por fecha (1° quincena de Agosto)

Desde el 01/08/2011 hasta el 15/08/2011 el máximo detectado ha sido de 128 dispositivos el día 09/08.

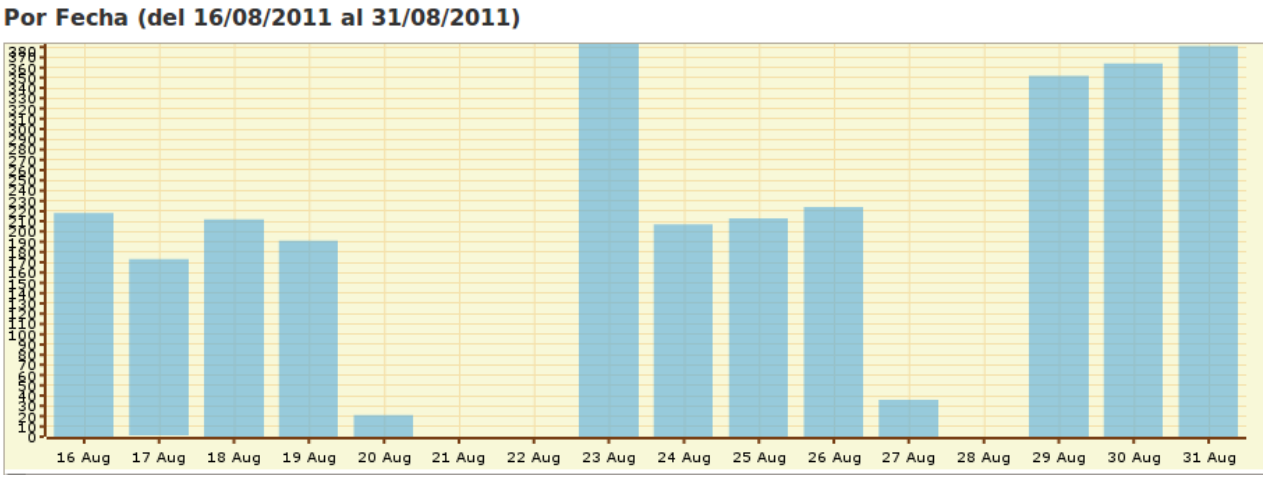


Fig 40: Dispositivos encontrados por fecha (2° quincena de Agosto)

Desde el 16/08/2011 hasta el 31/08/2011 el máximo detectado ha sido de 383 dispositivos el día 23/08.

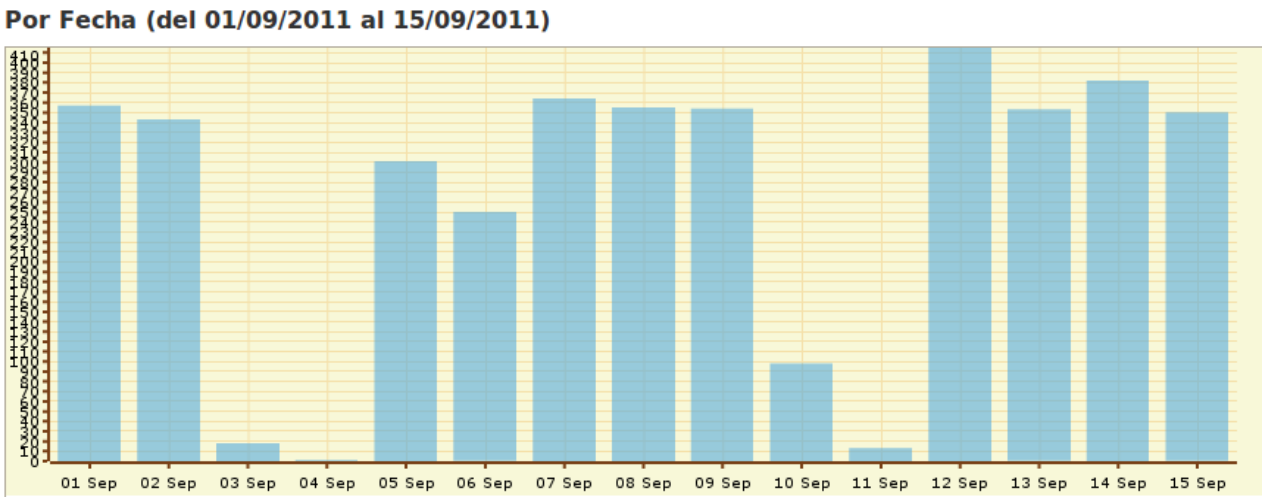


Fig 41: Dispositivos encontrados por fecha (1° quincena de Septiembre)

Desde el 01/09/2011 hasta el 15/09/2011 el máximo detectado ha sido de 415 dispositivos el día 12/09.

Se puede observar un notable crecimiento en las últimas dos quincenas debido también a la colocación de nuevos sensores en la entrada de calle 48.

6.3.9 Dispositivos detectados por su tipo

El sistema también permite diferenciar los distintos tipos de dispositivos detectados:

Tipo (clase mayor/clase menor)	*Dispositivos*	*Porcentaje*
Phone/Cellular	10017	77,95%
Phone/Smartphone	2467	19,20%
Phone/Cordless	174	1,35%
Miscellaneous/Uncategorized	88	0,68%
Computer/Uncategorized	46	0,36%
Computer/Laptop	26	0,20%
Computer/Hanheld PC/PDA (clam shell)	19	0,15%
Audio/Video/Uncategorized	9	0,07%
Computer/Desktop workstation	3	0,02%
Computer/Palm sized PC/PDA	1	0,01%
TOTAL	**12850**	**100,00%**

Tabla 12: Tipos de dispositivos detectados

Se puede apreciar que la mayoría de los dispositivos han sido celulares y que aún siguen predominando aquellos que no son Smartphones, o bien, si su cantidad es mayor, los estudiantes no suelen utilizarlos con el Bluetooth activado. Nótese también el porcentaje de teléfonos inalámbricos detectados con tecnología Bluetooth, los cuales son aprovechados para conectarlos a celulares y contestar llamadas recibidas desde los mismos. El gráfico que presenta el sistema es el siguiente:

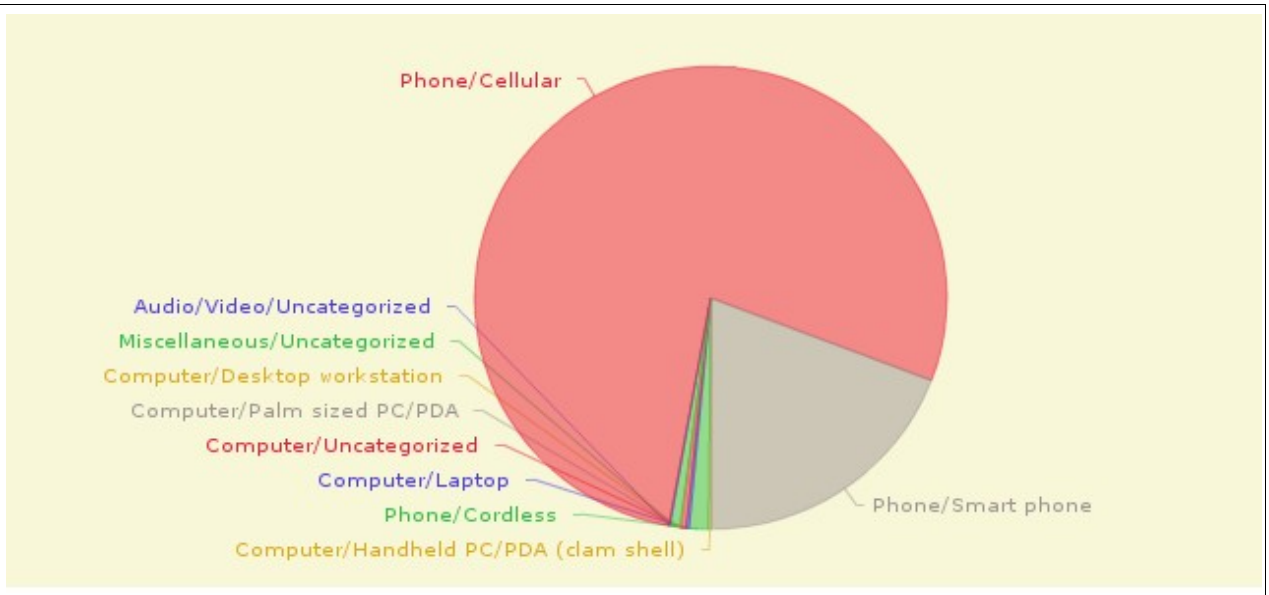


Fig 42: Dispositivos detectados por tipo

6.3.10 Dispositivos detectados por lugar

Según los datos relevados, el Subsuelo fue el lugar donde más dispositivos se detectaron (57,75%), seguido por el Hall Principal (24,39%) y por último la Entrada de calle 48 (17,86%). Sin embargo, estos datos no representan la realidad en el mismo período, ya que en el Hall Principal se dejó de detectar una vez que se trasladaron los sensores al Subsuelo y más adelante se agregaron los sensores de la Entrada de calle 48, por lo que hace suponer que este último lugar, con el correr del tiempo, superaría en proporción al resto de los lugares.

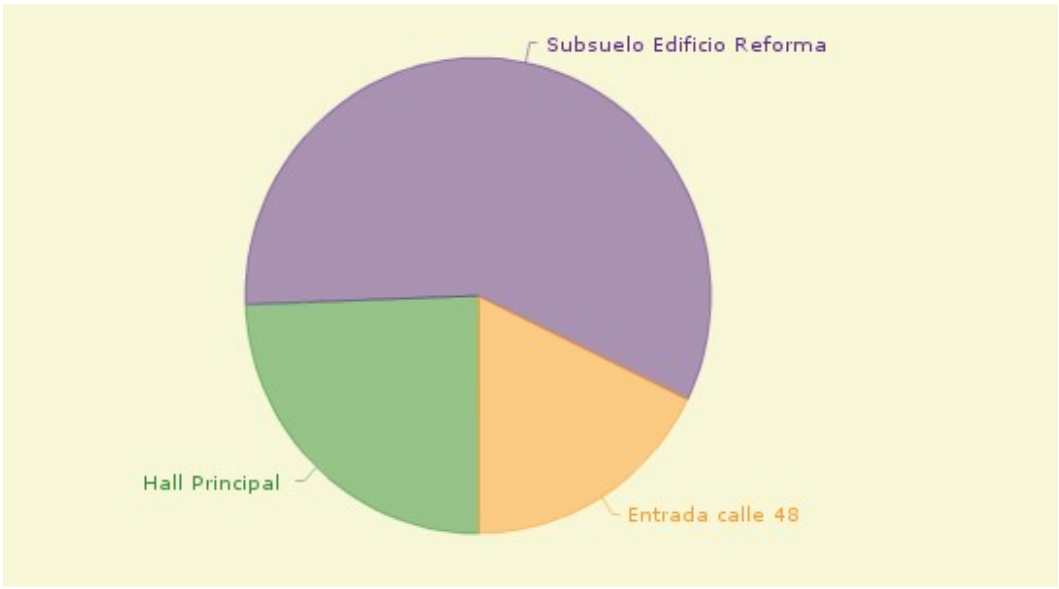


Fig 43: Dispositivos encontrados por lugar

A raíz de estos datos, se realizaron estadísticas a partir de la puesta en producción de los sensores en la Entrada de calle 48 (desde el 29/08/2011 hasta el 16/09/2011) para calcular cuántos dispositivos han sido detectados en total en la Entrada, en el Subsuelo, en ambos lugares y en cada lugar sin contar aquellos que también se detectaron en el otro. Los resultados son mostrados en la siguiente tabla:

Lugar	*Dispositivos únicos*	*Dispositivos recurrentes*
Subsuelo	386	3194
Entrada calle 48	508	1857
Subsuelo y Entrada calle 48	236	2626
Subsuelo sin Entrada calle 48	150	568
Entrada calle 48 sin Subsuelo	272	593

Tabla 13: Cantidad detallada de dispositivos por lugar

Con dispositivos únicos se refiere a los dispositivos encontrados sin contar sus repeticiones, y con recurrentes teniéndolas en cuenta. El gráfico asociado es el siguiente:

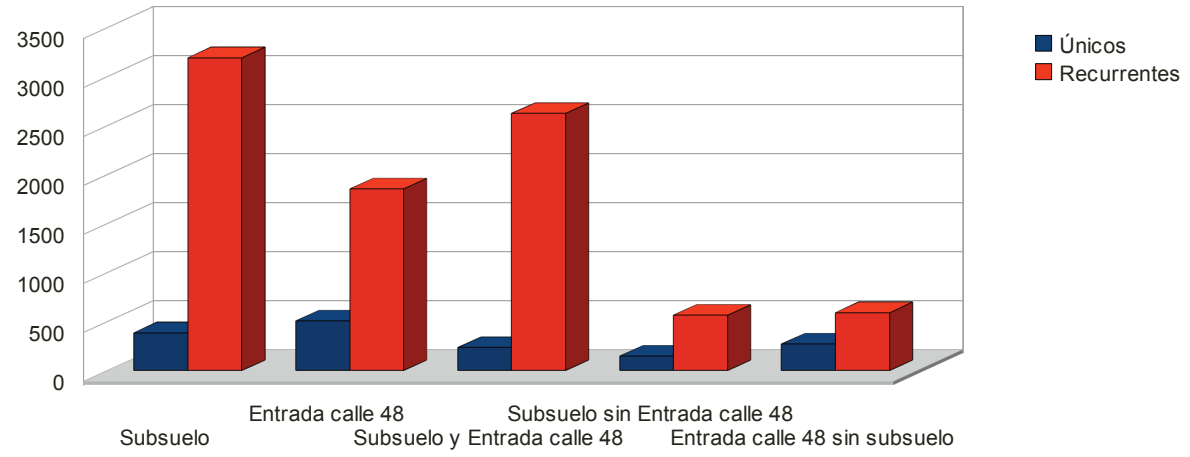


Fig 44: Dispositivos detallados por lugar

6.3.11 Nombres de los dispositivos

Realizando un análisis de los nombres descubiertos durante la detección de dispositivos, se pudo verificar que más de la mitad modificaron el nombre que viene de fábrica (53%), pero una gran parte (47%) no lo hizo. Esto se puede deber a que mucha gente no se da cuenta que tiene el detector Bluetooth encedido y no tienen interés en cambiarlo. De todas maneras más de la mitad lo hizo y son posibles envíos con éxito.

El gráfico que lo demuestra es el siguiente:

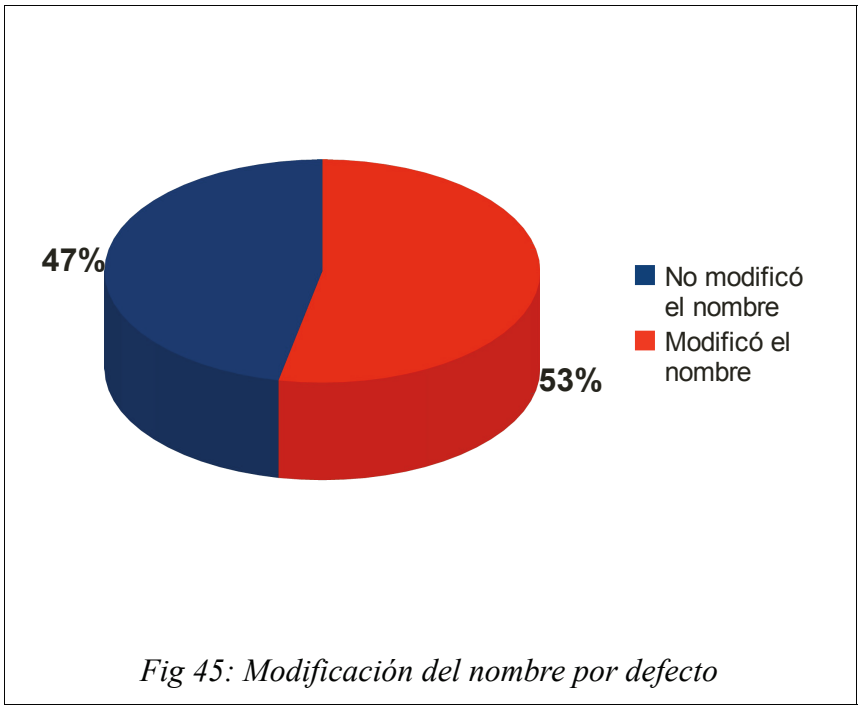


Fig 45: Modificación del nombre por defecto

6.3.12 Análisis de los datos relevados

Como primer conclusión acerca de todos los datos relevados, se puede observar que hubo un gran incremento de detecciones a lo largo de las pruebas con la colocación de nuevos sensores en diferentes lugares, y que existe una correlación real con las fechas respecto a los fines de semana (con menor o casi nula detección de dispositivos) y el receso invernal (sin detecciones).

Otra conclusión a la que se ha arribado es que mucha gente que suele transitar el subsuelo ha ingresado por la entrada de calle 48, pero no todos los que entraron por ahí se han dirigido al subsuelo. Pero en el subsuelo también se detectaron dispositivos que no fueron detectados en el ingreso de calle 48, por lo que se supone que entraron por el ingreso de calle 49 o son dispositivos que fueron encendidos una vez dentro del edificio.

7 Conclusiones

Como se ha detallado a lo largo del trabajo, las decisiones tomadas, elecciones de las tecnologías y arquitectura, el desarrollo del prototipo, etc. fueron cuestiones estudiadas, y los diversos problemas presentados fueron afrontados y solucionados.

Se siguió una misma política, la utilización del Software Libre, lo cual permite abrir muchas puertas y sobre todo hace posible su implementación en diversos lugares sin pensar en licencias pagas, ni del sistema operativo ni de ninguna de las librerías, sistemas ni servicios involucrados. La comunidad de Software Libre aporta día a día nuevas herramientas y demuestra que el esfuerzo colectivo y colaborativo es sin dudas un éxito no solo en la producción de software sino en todos los aspectos posibles. Es por esto que en el desarrollo de Blue4AS se ha decidido compartirlo como Software Libre bajo una licencia GPL para que pueda ser extendido y adaptado para diferentes necesidades; es un pequeño aporte.

Los resultados obtenidos superan ampliamente las expectativas, y si bien lo que se logró es bajo la carátula de “Prototipo”, queda claramente demostrado que se puede poner en producción con total éxito para recolectar información muy importante y útil sobre la circulación en el recinto monitoreado. Además es sumamente útil el envío de información sobre lo que sea de importancia para ese espacio, en este caso, la seguridad, el apoyo y servicios para personas con capacidades diferentes. Pero el resultado obtenido es una solución totalmente genérica aplicable a diversas implementaciones incluso dentro de un mismo ámbito. Así como ha sido demostrado que el prototipo se puede poner en producción, es evidente que quedan también muchas cosas por hacer, como lo plasmado en el Capítulo 5 (Otras aplicaciones y trabajos futuros), como así también realizar pruebas con otro tipo de hardware y estudiar su comportamiento. También sería de suma importancia la expansión del módulo de estadísticas de la administración remota (sistema Web) ya que cada implantación podría requerir de diferentes análisis. En este aspecto lo más importante es el conocimiento que adquiere la organización -información almacenada-, y a partir del mismo se pueden generar infinidad de consultas y estadísticas.

La idea principal fue realizar un sistema de envío de cualquier tipo de dato a cualquier tipo de dispositivo que disponga de Bluetooth sin la necesidad de que el mismo

posea una aplicación cliente para detectarlo y recibirlo, y si bien existen algunos casos que es inevitable realizarlo sin ella, en la mayoría de ellos se ha logrado satisfactoriamente. Otro inconveniente surgido fue el emparejamiento de los dispositivos, algo muy común en la comunicación vía Bluetooth por seguridad, y se logró obtener un emparejamiento automático con los dispositivos que así lo requieran mediante algún código predeterminado, como por ejemplo “1234”, del lado del transmisor, es decir, que es necesario que el dispositivo receptor ingrese el código para poder recibir los datos a enviar, algo imposible de evitar. Si bien se ha logrado dicha automatización, el emparejamiento se debe realizar por cada sensor utilizado para la transmisión, ya que en el dispositivo queda almacenada la dirección MAC del mismo y cada sensor tiene una MAC diferente, por lo que requiere un nuevo emparejamiento.

Si bien al principio se creyó que la mayoría de las personas no tenían habilitado el Bluetooth de sus celulares de manera predeterminada, ha sido de una gran sorpresa notar que sí y que incluso mucha gente siquiera modificó el nombre del dispositivo que viene de fábrica. Las distintas políticas pensadas para el envío se basaron en ello y al utilizarse la política de lista negra en la prueba de campo, no fue necesario hacer publicidad del prototipo, ya que la expansión de su funcionamiento se logró más de boca en boca a partir de la gente que le llegó y aceptó por curiosidad, lo que permitió realizar un seguimiento detallado de lo ocurrido sin previo aviso.

Respecto a los objetivos planteados (orientación y servicios al usuario y conocimiento de la organización), el desarrollo del prototipo y la posterior implantación en la Facultad de Ciencias Jurídicas y Sociales le permitió, por un lado a la Unidad Académica mencionada, obtener datos nunca antes relevados ni imaginados. Por primera vez en su historia, se pudieron determinar diferentes flujos de circulación, permanencia por sectores, estimar las franjas horarias de mayor y menor circulación, cantidad de ingresos por su acceso principal, cantidad de envíos exitosos, estimaciones por clase de dispositivos y tipo de información enviada, etc. Esta información podría servir para tomar diferentes decisiones, plantear cambios en los accesos, escaleras, rampas, y diferentes cuestiones edilicias como de índole informativas para los usuarios, brindando información accesible por todos, audio para los no videntes, textos e imágenes de mapas para los hipoacúsicos o aquellas que posean sordera. Es tan importante el formato utilizado como el contenido de la información; se brindaron mapas de localización haciendo énfasis en las salidas de emergencia, ubicación de matafuegos, escaleras, ascensores y rampas de

acceso para discapacitados.

Por último y como conclusión final se puede decir que al ser un proyecto de bajo costo de equipamiento y mantenimiento, es muy factible su implantación en lugares con escasos recursos y que puede brindar información muy importante para la organización, pero sobre todo el servicio brindado a los usuarios puede ser de una importancia superlativa para el que recibe la información además de percibir una real inclusión en el sistema. Darle accesibilidad va a depender de la configuración y de los datos que se envíen por parte de la organización, pero es un deber de toda la sociedad colaborar con ello. Como tiene que haber rampas, indicadores braile y señalética acorde a diferentes formas de percepción en distintos lugares, es responsabilidad de los productores de software y tecnologías tener en cuenta estos aspectos y basarse en un principio que debería ser de común denominador por todas las partes: “La información **debe** estar disponible para **todos**”.

“The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect”.

– Tim Berner Lee, W3C

– Director & Inventor of the World Wide Web.

Índice de Figuras

Fig 1: Modelo de entrada adicional de datos....14
Fig 2: Modelo de elemento que modifica la entrada de datos....15
Fig 3: Modelo de información que vuelve al usuario....15
Fig 4: Modelo de disparador de eventos....15
Fig 5: Configuración de ejemplo de las componentes en herramientas de contexto....20
Fig 6: Pila de protocolos Bluetooth....24
Fig 7: Perfiles Bluetooth....28
Fig 8: Piconet....30
Fig 9: Scatternet....31
Fig 10: Categorías de software....46
Fig 11: Cliente (dongle station) con 3 sensores Bluetooth conectados a un Hub USB de 7 puertos. 57
Fig 12: Proceso de descubrimiento de dispositivos....59
Fig 13: Aplicar política de broadcasting....61
Fig 14: Enviar datos a un dispositivo....63
Fig 15: Diagrama de clases del Sistema Ubicuo....67
Fig 16: Configuración general....86
Fig 17: Diagrama de clases del Sistema Remoto....88
Fig 18: Agregar un contacto....91
Fig 19: Agregar un evento....93
Fig 20: Dispositivos Encontrados detallados....95
Fig 21: Estadísticas de dispositivos enviados por Fecha....96
Fig 22: Estadísticas de dispositivos enviados por Hora....96
Fig 23: Estadísticas de dispositivos enviados por Lugar....97
Fig 24: Estadísticas de dispositivos enviados por Tipo....97
Fig 25: Estadísticas de datos enviados por Tipo....98
Fig 26: Estadísticas de un dato particular enviado por Fecha....99
Fig 27: Vcard en Thunderbird....115
Fig 28: Plano subsuelo....116
Fig 29: Dispositivos nuevos y recurrentes....122
Fig 30: Promedio de dispositivos por día....125

Fig 31: Cantidad de dispositivos por tiempo de permanencia (Subsuelo).....126
Fig 32: Cantidad de dispositivos por tiempo de permanencia (Entrada calle 48).....126
Fig 33: Cantidad de envíos totales.....127
Fig 34: Datos aceptados y rechazados.....128
Fig 35: Cantidad de envíos con error por tipo.....128
Fig 36: Datos enviados por lugar.....129
Fig 37: Datos enviados por tipo de dato.....130
Fig 38: Dispositivos detectados por hora.....131
Fig 39: Dispositivos encontrados por fecha (1° quincena de Agosto).....132
Fig 40: Dispositivos encontrados por fecha (2° quincena de Agosto).....132
Fig 41: Dispositivos encontrados por fecha (1° quincena de Septiembre).....133
Fig 42: Dispositivos detectados por tipo.....134
Fig 43: Dispositivos encontrados por lugar.....135
Fig 44: Dispositivos detallados por lugar.....136
Fig 45: Modificación del nombre por defecto.....136

Índice de Tablas

Tabla 1: Potencia y rango por clase de sensores.....23
Tabla 2: Combinaciones posibles para el descubrimiento y conexión de un dispositivo Bluetooth. .33
Tabla 3: Potencias máximas permitidas para Bluetooth.....41
Tabla 4: Resumen de los límites de exposición recomendados por la ICNIRP [20].....43
Tabla 5: Especificación del campo de clase de Dispositivo/Servicio.....75
Tabla 6: Clasificación de las clases de dispositivos según los bits.....75
Tabla 7: Subclasificación de dispositivos de la clase Computer.....76
Tabla 8: Subclasificación de dispositivos de la clase Phone.....77
Tabla 9: Promedio de detecciones únicas y totales por semana.....124
Tabla 10: Tipos de errores de envío detectados.....129
Tabla 11: Cantidad de dispositivos totales detectados por hora.....131
Tabla 12: Tipos de dispositivos detectados.....133
Tabla 13: Cantidad detallada de dispositivos por lugar.....135

8 Bibliografía

[1] Barbara Pernici (Ed.), *Mobile Information Systems. Infrastructure and Designfor Adaptivity and Flexibility*, Springer. 2006.

[2] Juergen Sieck, Michael A. Herzog, "Mobile Information Systems and Mobile Learning". FHTW Berlin, University of Applied Sciences Department of Applied Computer Science, Research group »INKA«.

[3] Mark Weiser and John Seely Brown, "The coming age of Calm Technology", 05-Oct-1996. [Online]. Available: http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm.

[4] Asociación de Técnicos de Informática, "Computación Ubicua", *Upgrade Novática*, nº. 153, Sep-2001.

[5] Anind K. Dey, "Understanding and Using Context".

[6] David Sainz González, *Un análisis sobre aplicaciones distribuidas dependientes del contexto*. UNIVERSIDAD DEL PAÍS VASCO, 2009.

[7] Cecilia Challiol, "Asistencia sensible al contexto en aplicaciones de Hipermedia Física". [Online]. Available: http://www.lifia.info.unlp.edu.ar/es/difusion/20070816.htm.

[8] B. Schilit, N. Adams, y R. Want, "Context-Aware Computing Applications", *IN PROCEEDINGS OF THE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS*, pág. 85--90, 1994.

[9] M. Baldauf, S. Dustdar, y F. Rosenberg, "A survey on context-aware systems", *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 2, nº. 4, págs. 263-277, 2007.

[10] Anind K. Dey, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications". .

[11] Albert S. Huang, *Bluetooth Essentials for Programmers*. Cambridge University Press, 2007.

[12] David Kammer, *Bluetooth Application Developer's Guide: the Short Range Interconnect Solution*, Syngress Publishing, Inc. 2002.

[13] Christian Gehrmann, *Bluetooth Security*. Artech House, 2004.

[14] "UIT: Comprometida para conectar el mundo". [Online]. Available: http://www.itu.int/es/pages/default.aspx.

[15] "ITU Frequently asked questions". [Online]. Available: http://www.itu.int/ITU-R/terrestrial/faq/index.html#g013.

[16] "Article 15 - Interferences". [Online]. Available: http://life.itu.int/radioclub/rr/art15.htm.

[17] "Comisión Europea". [Online]. Available: http://ec.europa.eu/index_es.htm.

[18] "Federal Communications Commission (FCC) Home Page". [Online]. Available: http://www.fcc.gov/.

[19] "Comisión Nacional de Comunicaciones". [Online]. Available: http://www.cnc.gov.ar/ciudadanos/espectro/rni_efectos.asp.

[20] "OMS | ¿Qué son los campos electromagnéticos?" [Online]. Available: http://www.who.int/peh-emf/about/WhatisEMF/es/index4.html.

[21] Richard M. Stallman, *Software Libre para una sociedad libre*, Traficantes de Sueños. 2004.

[22] "The GNU Operating System". [Online]. Available: http://www.gnu.org/.

[23] Revista Virtual: "Negocios de Seguridad", "DVR sobre plataforma Linux".

[24] Raúl González Duque, *Python para todos*. España: .

[25] Adrian Holovaty, *The Definitive Guide to Django*. Apress, 2008.

[26] "Instituto Nacional de Tecnología Industrial - Argentina". [Online]. Available: http://www.inti.gob.ar/sabercomo/sc23/inti9.php